Master's Thesis

Development of an Environment Framework for a Modular Vehicle Simulation System

Xavier Bas Ferrer

Institute Director

Univ.-Prof. Dr.techn. Wolfgang Hirschberg

Advisor

Andrés Eduardo Rojas Rojas, M.Sc.

Institute of Automotive Engineering

Member of the Frank Stronach Institute

GRAZ UNIVERSITY OF TECHNOLOGY

School of General Engineering of Barcelona

Technical University of Catalonia

Graz, Austria

October 2011

Abstract

Recent vehicle developments are forced by continuously decreasing development times. To reach this aim virtual methods are necessary. Concerning the research area of vehicle dynamics, virtual analyses play an important role during the whole development process of a vehicle. Vehicle dynamics simulations using different manoeuvres, models and environments become a key component. This important role is even more significant for the early design phase of the vehicle, where only few and uncertain parameters are available.

The present thesis expands the simulation platform MOVES^2, designed for modular vehicle dynamic simulations during the early design phase, at the Institute of Automotive Engineering at Graz University of Technology. The presented expansion implements a set of modules to generate arbitrary roads within arbitrary scenarios, to animate the simulated vehicle motion within a virtual reality world based on the generated scenario, to define the desired trajectory for the simulation and to configure the modelling complexity of the model.

To ensure a high level of user-friendliness, the Graphical User Interface of MOVES^2 manages the required data from these modules, by creating, editing and deleting it.

Acknowledgments

First I would like to thank international departments of TU Graz and ETSEIB, as well as to the whole FTG institute, with special mention to Andrés Eduardo Rojas Rojas, M.Sc., and Univ.-Prof. Dr.techn. Wolfgang Hirschberg, for giving me the opportunity to write my thesis in Graz. I am strongly grateful for taking me on trust, it was an enriching and fantastic experience.

I would like to express my gratitude to our research team formed by Andrés Eduardo Rojas Rojas, M.Sc., Dipl.-Ing. Haymo Niederkofler, Josef Dürnberger, and me, Xavier Bas Ferrer. It was great to work in it; I received a lot from them. Thank you Andrés for your continuously feedback and your nearly full-time availability. Haymo, thanks for your advises and for the good and funny time in the office. And Josef, thank you for your never-ending jokes, the *Yo* therapy was perfect after some working hours.

Many thanks for create a nice environment in the FTG institute and the good moments in our multifunctional library. Severin Stadler, Jorge Enrique Wilson Aponte, David Böhm, Gernot Frener, Harald Kraus, and Dipl.-Ing. Daniel Wallner. I had a great time there with you. Also thanks Severin for assisting me with CATIA software, and thanks Daniel for giving me guidance on the writing of this document.

I would like to especially thank my family for the continuous support, and special thanks to Maria Ferrer Bonet for pushing me to give up bad habits in writing. Finally, I want to thank Andrea Hom Díaz for the inexhaustible motivation during my writing period.

TABLE OF CONTENTS

1. INTRODUCTION 1

1.1. DEVELOPMENT PROCESS IN THE AUTOMOTIVE INDUSTRY 1

1.2. MOTIVATION 2

1.3. STATE OF THE ART 2

1.4. WORK CONTENT 3

1.5. THESIS STRUCTURE 4

2. SIMULATION ENVIRONMENT MOVES^2 5

2.1. MOTIVATION TO DEVELOP MOVES^2 5

2.2. USABILITY SCOPE 6

2.3. STRUCTURE, SIMULATION PROCESS, AND MODULARITY OF MOVES^2 6

2.4. DRIVER-VEHICLE-ENVIRONMENT SYSTEM 8

2.5. MANOEUVRE-BUILDING MODULE 9

3. THE THREE-DIMENSIONAL ROAD 12

3.1. COORDINATE SYSTEMS 12

3.2. GEOMETRIC ROAD DESIGN 14

3.2.1. Horizontal alignment 15

3.2.2. Vertical alignment 17

3.3. OPENCRG® 18

3.3.1. The CRG-road, main features 20

3.3.2. Coordinate systems, transformations 24

3.4. SYNERGIES OPENCRG® – MOVES^2 29

3.4.1. Road perturbations: obstacles 30

3.4.2. Road roughness 31

3.4.3. Friction coefficient 32

3.5. THE USAGE OF 3-D ROADS IN 2-D VEHICLE MODELS 32

4. ANIMATION MODULE 34

4.1. VIRTUAL REALITY MODELLING LANGUAGE 34

4.1.1. VRML coordinate system 34

4.1.2. The VRML file 35

4.2. THE VRML SCENE 35

4.2.1. Vehicle shape 37

4.2.2. Scenario world *37*

4.2.3. Unanimated objects *38*

4.3. ANIMATION PERFORMANCE 39

5. SCENARIO-BUILDING MODULE 40

5.1. SCENARIO MODULARITY 40

5.2. SCENARIO MANAGEMENT 41

6. MODEL MANAGEMENT TOOL 46

7. CONCLUSIONS 50

BIBLIOGRAPHY 52

A. THE GUI SCENARIO PANELS 55

B. THE GUI ROAD PANEL 62

C. EXPORT CATIA MODELS TO THE VRML FORMAT 68

D. ANIMATION WITH MULTIPLE CARS 71

TABLE OF FIGURES

FIGURE 1.1. THE AUTOMOTIVE DESIGN STAGES [HIR_10]. 1

FIGURE 2.1. EXAMPLE OF AN INNOVATIVE CONCEPT, THE ECORNER-MODULE, WITH INTEGRATED FUNCTIONALITIES WITHIN THE RIM [ROJ_11A]. 5

FIGURE 2.2. STRUCTURE OF THE MOVES^2 AND SIMULATION PROCESS CARRIED OUT IN IT [ROJ_11A]. 7

FIGURE 2.3. MODEL MODULARITY IN THE MOVES^2 [ROJ_11A]. 8

FIGURE 2.4. DRIVER-VEHICLE-ENVIRONMENT SYSTEM [ROJ_11A]. 9

FIGURE 2.5. CUSTOMISED TRAJECTORY USING THE FIRST OPTION OF THE MANOEUVRE-BUILDING MODULE: ROAD DATABASE-LIBRARY. TRAJECTORY OF A CIRCUIT SHIFTED 5,5 METERS TO THE RIGHT. 10

FIGURE 2.6. CUSTOMISED TRAJECTORY USING THE SECOND OPTION OF THE MANOEUVRE-BUILDING MODULE: TRAJECTORY DATABASE-LIBRARY. TRAJECTORY OF THE STANDARDIZED DOUBLE LANE-CHANGE [ISO_99] GENERATED WITH THE SIGNAL BUILDER BLOCK. 10

FIGURE 2.7. CUSTOMISED TRAJECTORY USING THE THIRD OPTION OF THE MANOEUVRE-BUILDING MODULE: TRAJECTORY FUNCTIONS. SINUSOIDAL TRAJECTORY WITH 10 M. OF AMPLITUDE, 50 M. OF PERIOD AND $-\pi/2$ RAD. OF INITIAL PHASE. 11

FIGURE 2.8. DESIRED AND REAL TRAJECTORIES IN A SEVERE DOUBLE-LANE CHANGE TEST [ISO_99] DRIVEN AT 18 M/S. SCREENSHOT OF THE VISUALIZATION FEATURE IMPLEMENTED IN MOVES² AS PART OF THE PRESENT WORK. 11

FIGURE 3.1. VEHICLE POSITION MEASURED IN x-y AND u-v COORDINATES. SCREENSHOT OF THE VISUALIZATION FEATURE IMPLEMENTED IN THIS WORK. 13

FIGURE 3.2. TRAJECTORY COMPOSED BY A TANGENT FOLLOWED BY A CURVE WITH TRANSITION, WITH THE SUPERPOSITION OF ITS EULER SPIRAL (A) AND THE DRIVEN DISTANCE OF THE TRAJECTORY VS. ITS CURVATURE (B). 16

FIGURE 3.3. METHODS TO DISTRIBUTE BANKING ALONG THE TRANSITION CURVE [FIT_07]. 17

FIGURE 3.4. ROAD PROFILE WITH A CREST VERTICAL CURVE WITH 10% OF MAXIMUM $slope$ RATE. SCREENSHOT OF THE VISUALIZATION FEATURE IMPLEMENTED IN THIS WORK. 18

FIGURE 3.5. A CURVED REGULAR GRID ROAD DESCRIPTION WITH ITS CURVED REFERENCE LINE – CRL – HIGHLIGHTED. 19

FIGURE 3.6. GEOMETRICAL REQUIREMENTS OF OFFSET LINES FROM THE CURVED REFERENCE LINE IN A CURVED REGULAR GRID [DAI_10] [VIR_11]. 20

FIGURE 3.7. PLOTS FROM THE INFORMATION OF A CURVED REFERENCE LINE. 21

FIGURE 3.8. ROAD DESCRIBED IN CARTESIAN COORDINATES WITH THE MINIMAL INFORMATION TO BE GENERATED. 21

FIGURE 3.9. GEOMETRIC RELATIONS OF THE SURROUNDINGS OF A POINT P BEING TRANSFORMED FROM CARTESIAN TO SURFACE COORDINATES [DAI_10]. 25

FIGURE 3.10. OPENCRG© TRANSFORM EVALUATION TOOL x-y TO u-v FLOW DIAGRAM [VIR_11]. 25

FIGURE 3.11. GEOMETRIC RELATIONS OF THE SURROUNDINGS OF A POINT P BEING TRANSFORMED FROM SURFACE COORDINATES TO CARTESIAN COORDINATES [DAI_10]. 27

FIGURE 3.12. OPENCRG© TRANSFORM EVALUATION TOOL u-v TO x-y FLOW DIAGRAM [VIR_11]. 28

FIGURE 3.13. ROAD WITH BUMPS COMPOSED OF A FLAT AND STRAIGHT ROAD AND A DOUBLE BUMPED PERTURBATION PROFILE EXTRUDED IN v DIRECTION. CREATED WITH THE OBSTACLE-BUILDER DEVELOPED IN THIS WORK. 31
FIGURE 3.14. ROAD WITH ROUGHNESS CATEGORY C. 31
FIGURE 3.15. SCHEMATIC μ-SLIP CURVES FOR DIFFERENT ROAD CONDITIONS, WITH EXAGGERATED INITIAL SLOPE DIFFERENCES [GUS_98]. 32
FIGURE 3.16. WHEEL CONTACTS FOR A SPECIFIED TRAJECTORY IN A THREE-DIMENSIONAL ROAD. 33
FIGURE 3.17. WHEEL CONTACTS FROM THE ROAD SHOWN IN FIGURE 3.16. 33
FIGURE 4.1. VRML COORDINATE SYSTEM [MAT_10]. 35
FIGURE 4.2. THE VRML FILE HIERARCHY OF THE VRML SCENE 36
FIGURE 4.3. ANIMATED OBJECTS DURING THE ANIMATION CODED WITHIN THE VRML SCENE. SCREENSHOT OF THE VISUALIZATION FEATURE IMPLEMENTED IN THIS WORK. 36
FIGURE 4.4. VEHICLE SHAPE DESIGNED IN CATIA® V5 AT THE INSTITUTE OF AUTOMOTIVE ENGINEERING AT GRAZ UNIVERSITY OF TECHNOLOGY. 37
FIGURE 4.5. SCENARIO WORLD OF A SPLIT-MU MANOEUVRE WITH LANE LINES AND THE ROAD SECTION SIDE OF THE LOW μ COLOURED. SCREENSHOT OF THE VISUALIZATION FEATURE IMPLEMENTED IN THIS WORK. 38
FIGURE 4.6. THE PREVIEW OF LIGOS V·REALM BUILDER FOR THE UNANIMATED OBJECT TREE. 38
FIGURE 5.1. THE SCENARIO LIBRARY AND ITS LINKED SUB-LIBRARIES. SPECIFIC PARTS OF THE GUI OF MOVES^2 CAN ADJUST THE ROAD, MANOEUVRE AND ANIMATION SPECIFICATIONS. 41
FIGURE 5.2. FLOW DIAGRAM OF THE SCENARIO PRE-PROCESSING FUNCTION. 42
FIGURE 5.3. THE INTERACTION OF THE SCENARIO PRE-PROCESSING FUNCTION WITH THE DATABASE-LIBRARIES. 43
FIGURE 6.1. HIERARCHY OF SOME BLOCK CHOICES ACCORDING TO THE MODEL IMPLEMENTED IN MOVES^2. 47
FIGURE 6.2. REARRANGED HIERARCHY OF SOME BLOCK CHOICES ACCORDING TO THE MODEL IMPLEMENTED IN MOVES^2. 48
FIGURE 6.3. A PART OF THE MODEL MANAGEMENT PANEL SHOWING AN EXAMPLE OF THE REORGANIZED HIERARCHY OF COMPONENTS AND SYSTEMS RELATED WITH WHEELS. 49
FIGURE A.1. THE SCENARIO MASTER TOOLBAR. 55
FIGURE A.2. SCRIPT MANAGEMENT SUB-PANEL TO MODIFY THE CONTENT OF LIBRARIES AND SUB-LIBRARIES FROM THE PARAMETERS LIBRARY. 55
FIGURE A.3. CUSTOMIZE PANEL FOR A DOUBLE LANE-CHANGE MANOEUVRE. 56
FIGURE A.4. ROAD / OBSTACLE PANEL FOR A BUMPED ROAD. 57
FIGURE A.5. ROAD / MU PANEL FOR A SPLIT-MU MANOEUVER. 58
FIGURE A.6. TRAJECTORY PANEL FOR A SEVERE DOUBLE LANE-CHANGE MANOEUVRE. 59
FIGURE A.7. ANIMATION: OBJECTS / CAR PANEL FOR A DOUBLE LANE-CHANGE MANOEUVRE. 60
FIGURE A.8. ANIMATION: LINES PANEL FOR A DOUBLE LANE-CHANGE MANOEUVRE. 61
FIGURE A.9. THE SCHEMATIC ROAD PLOT PANEL FOR A DOUBLE LANE-CHANGE MANOEUVRE. 61
FIGURE B.1. THE GUI ROAD PANEL WITH THE VOID ROAD FILE SELECTED. 63
FIGURE B.2. EXAMPLE OF A ROAD ALIGNMENT CONSTRUCTION. GENERAL DATA FULFILLED. 63
FIGURE B.3. EXAMPLE OF A ROAD ALIGNMENT CONSTRUCTION. INTRODUCING CURVATURE DEFINITION I. 64
FIGURE B.4. EXAMPLE OF A ROAD ALIGNMENT CONSTRUCTION. INTRODUCING CURVATURE DEFINITION II. 64

FIGURE B.5. 3-D PLOT OF THE ROAD EXAMPLE. CURVATURE SPECIFIED. 65
FIGURE B.6. EXAMPLE OF A ROAD ALIGNMENT CONSTRUCTION. INTRODUCING BANKING DEFINITION. 65
FIGURE B.7. 3-D PLOT OF THE ROAD EXAMPLE. CURVATURE AND BANKING SPECIFIED. 66
FIGURE B.8. EXAMPLE OF A ROAD ALIGNMENT CONSTRUCTION. INTRODUCING SLOPE DEFINITION. 66
FIGURE B.9. 3-D PLOT OF THE ROAD EXAMPLE. CURVATURE, BANKING, AND SLOPE SPECIFIED. 67
FIGURE C.1. VRML COMPATIBILITY DIALOG BOX. 68
FIGURE C.2. CATIA AND VRML COORDINATE SYSTEMS [MAT_10]. 69
FIGURE C.3. CATIA STANDARD VIEW OF THE PART, WHICH IS READY FOR THE EXPORTATION. 69
FIGURE C.4. THE PREVIEW OF LIGOS V·REALM BUILDER FOR THE EXPORTED OBJECT. 70
FIGURE C.5. THE EXPORTED CATIA MODEL WORKING WITHIN THE VRML SCENE. SCREENSHOT OF THE VISUALIZATION FEATURE IMPLEMENTED IN THIS WORK. 70
FIGURE D.1. EXAMPLE OF AN ANIMATION WITH TWO CARS ON SCENE. SCREENSHOT OF THE VISUALIZATION FEATURE IMPLEMENTED IN THIS WORK. 71

1. INTRODUCTION

This thesis enhances the environment framework of the existing simulation platform for vehicle dynamics named as MOdular VEhicle Simulation System, MOVES^2: A module to generate arbitrary three-dimensional roads, defined within three levels of detail together with the addition of friction coefficient, is implemented in this work. A manoeuvre-building module allows creating desired trajectories, for the vehicle simulation, with the *Signal Builder* tool or based on existing roads or user-defined functions. A configurable tool to animate the simulated vehicle motion is also introduced in this work. These features are grouped in a modular scenario capable to change, edit and create its pieces. Moreover, this work adds a tool to configure the modelling complexity of the systems and components within the vehicle model of MOVES^2.

The present project is the result of ten months of intensive work at the Institute of Automotive Engineering at Graz University of Technology together with Magna Powertrain and Magna Steyr.

This chapter presents the research project, beginning with the background and motivation of the project, followed by an overview of the initial development stage of the simulation platform and the reasons why the present research has been developed. Finally, the chapter concludes with a description of the thesis structure.

1.1. Development process in the automotive industry

The existing vehicle market system – extremely competitive – together with the pressure made by the competitors and the demanding customers forced the car manufacturers to reduce the vehicle development time, while improving the quality product. The product improvement has been based on increase safety, comfort and on implement new functionalities on the vehicle [ROJ_11a].

Along the automotive history, manufacturers have studied their social, economic, and business situation. Consequently, they have been establishing accurate concepts and rules to reach the vehicle market requirements. I.e. Figure 1.1 shows the different main stages to manufacture a new vehicle, starting from its overall concept and finishing with the series production.

Figure 1.1. The automotive design stages [HIR_10].

Prototypes and virtual simulations are used during the entire automotive development process. Nevertheless, the tendency to reduce costs and development times has brought manufacturers to apply different strategies to different phases. During the early phases the vehicle development

evolves almost completely virtual. Then, in a subsequent phase, prototypes and in many cases no complete prototypes supported by virtual 3-D models and virtual components are used, besides, the models developed entirely in the computer. Finally, during the later development phases, real prototypes are used in most of the cases [FRÜ_04].

1.2. Motivation

Usually, in the early design phases of an innovative system research only few and uncertain parameters are available. During the Concept phase (see Figure 1.1), the main problems are the lack of data, and the different modelling complexity during the consecutive sub-phases. Therefore and to cope with these problems, the Institute of Automotive Engineering at Graz University of Technology has developed an approach that supports engineers during the early design phases of a project.

The vehicle models used in the Concept phase of a project of an innovative system present different complexities: for basic understanding of the systems only simple models are needed, but the analysis of a more realistic vehicle dynamic behaviour requires higher complexity models [ROJ_11a]. Moreover, results from different models or the results using different variable values are compared to gain knowledge and to support the decision-making process.

Inevitably, during the project development the vehicle model must handle a three-dimensional environment, treating the vehicle as an object with a three-dimensional motion. However, the required environment changes according to the vehicle development stage. Vehicle dynamics interests lie in the analysis of the body motion for specific manoeuvres, thus it is necessary to generate the correct simulation environment for each case. I.e. a test track configured to satisfy manoeuvre specifications may generate appropriate results to progress in the project development.

The configuration of a test track defines the road profile within three different levels: according to the macroscopic level, roads are configured as a succession of straights and vertical and horizontal curves. In a more detailed level of description, the road profile may contain perturbations (e.g. bumps, holes), and within the microscopic level, the roughness surface of a road is specified. Besides of the road profile, a test track can have sectors with different friction coefficients to be able to simulate some specific manoeuvres (e.g. split-μ manoeuvre [FOR_98]), and additionally it may require the usage of manoeuvre signals (cones, road signals, etc.) or other general objects (e.g. trees, houses).

1.3. State of the art

At the start point of this thesis, the simulation platform MOVES^2 was being developed together with a modular Graphical User Interface – GUI. To develop the approach a standard programing methodology was designed. Moreover, it was characterized as a modular, flexible and easily extendible approach, and a manual procedure could adjust the modelling complexity of the vehicle

model. Since the concept of MOVES^2 remains constant, these characteristics are described in Chapter 2.

MOVES^2 was capable to simulate within a two-dimensional environment; the entire driver-vehicle-environment system was already established according to the quarter and single track vehicle models. In that case, the road, which is the main component of the environment, was configured by a two-dimensional signal. The three-dimensional environment was being developed: although the three-dimensional equations of motion were being implemented in the model, the three-dimensional road was not integrated yet.

Therefore, the starting point of the present thesis was the addition of a road-builder in MOVES^2. Several companies offer programs and tools to generate roads, such as Cosin/3Droad [GIP_10], RoadXML© [CHA_10] or OpenCRG® [VIR_10]. OpenCRG© was the chosen tool to be implemented in the approach. OpenCRG© is an open source program written in MATLAB® and it contains a tool-suite that provides many features to generate, manage and evaluate three-dimensional roads [DAI_10].

1.4. Work content

The usage of a three-dimensional simulation environment opens interesting possibilities for realistic vehicle simulations. However, different simulation purposes may require different simulation scenarios. Hence, one of the goals of this thesis is to design and integrate a scenario-building module within the vehicle simulation environment. The designed scenario-building module is able to generate arbitrary scenarios in a modular way. It generates exchangeable specifications concerning the road profile viewed from a macroscopic standpoint, the road perturbations (e.g. holes, bumps or any signal), the road roughness, the friction of coefficient on the road, the desired trajectory for the driver controller and animation appearance configurations. An obstacle-builder has been developed to specify the road perturbations. Moreover, visualization tools can individually plot any specification of the scenario. The existing modular GUI of MOVES^2 handles this scenario-building module and its modular specifications.

As a result of the addition of an extra dimension in the environment, some parts of the existing approach have been modified, thus a manoeuvre-building module has been developed and integrated into MOVES^2 as a secondary goal of this thesis. This manoeuvre-building module is grouped in the scenario-building module and, as stated before, it can create the desired trajectory. The trajectory is based on the profile from a road, on a generated signal or on a user-defined function.

In order to compare different parameters at once, as well as improve the performance of no-experienced users, an animation module has been designed and integrated in MOVES^2. The animation module is also part of the scenario-building module. Through the animation specifications, a virtual reality world based on the simulation scenario can animate the simulated vehicle motion. Appearance configurations can highlight road perturbations, friction coefficient sections, the desired

trajectory and the road lanes. Moreover, they can add manoeuvre signals (e.g. cones) and other general objects (e.g. trees), and they can configure the vehicle shape of the simulation.

The third goal has been to expand the existing modular GUI of MOVES^2 in order to adjust in an easy way the complexity level required for the simulation. The vehicle model uses configurable subsystems to adapt the modelling complexity of its systems (e.g. damper actuator models) and components (e.g. tire models). The vehicle model is analysed and the found configurable subsystems are arranged and displayed on the GUI through tables.

At any case, these modules are developed in MATLAB/Simulink®.

1.5. Thesis structure

The following pages of this thesis are split into six chapters. Chapter 2 describes general aspects of the approach that have been developed. Chapter 3 contains detailed information about the three-dimensional environment needed to simulate, while Chapter 4 exposes the available virtual world used to visualize the simulation results. Chapter 5 explains how all the environment data is organized and managed. Chapter 6 introduces a tool to handle with the complexity level existing in the used full vehicle model.

This work concludes with Chapter 7 which summarises the present document and provides some outlook for further work.

2. SIMULATION ENVIRONMENT MOVES²

This chapter presents an overview of the MOdular Vehicle Simulation System – MOVES² – developed at the Institute of Automotive Engineering at Graz University of Technology [ROJ_11a]. The scenario- and the manoeuvre-building module, together with the animation module and the model management tool have been implemented in this simulation environment as stated in chapters 5 and 6.

2.1. Motivation to develop MOVES²

Concerning the research area of vehicle dynamics, during the early design phases of a vehicle development, projects involving innovative concepts must overcome the little available information problem. Moreover, especially in those stages, the level of uncertainty is high, while different development project lines and lots of variables should be taken into account. In this context, different modelling complexities are needed for different sub-phases within the early design phase, and for specific requirements. Eventually, it is complicated to find and adapt commercial modelling and simulation environments. For this reason, in those cases, a good alternative is to develop approaches using generic programming languages [ROJ_11a].

Figure 2.1 shows an example of an innovative concept, the eCorner-Module, which integrates, within the rim, different functions commonly present in a vehicle: propulsion, breaking, steering, and suspension system. The eCorner-Module opens new possibilities for vehicle design, where aerodynamics and crash load paths may be improved [NIE_10] [SIE_07]. Moreover, because of the simplicity of the new powertrain architecture, compared to conventional powertrain architectures, designs of car assembly lines can be leaner.

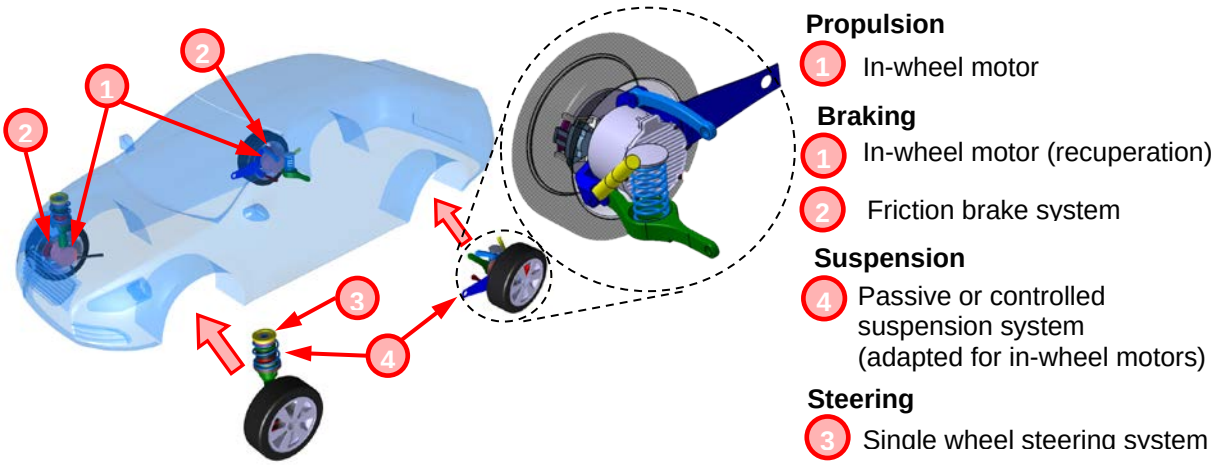


Figure 2.1. Example of an innovative concept, the eCorner-Module, with integrated functionalities within the rim [ROJ_11a].

The difficulties from early development phases of projects related with innovative concepts have motivated a research team, from the Institute of Automotive Engineering at Graz University of Technology, to develop a modular approach using the MATLAB/Simulink® programming language. This approach has been named MOdular Vehicle Simulation System, MOVES^2.

2.2. Usability scope

MOVES^2 is a simulation environment platform applicable during the stages from the early design phase. It uses a standardized programming methodology with a standardized way to manage the information. The overall concept of the approach characterizes its modular, flexible, and extendible properties: The features of MOVES^2 are internally organized by modules, keeping available its main parts at any case. New models, variables or features can be easily added following the standardized programming methodology. As a consequence of the exposed characteristics, it may support the knowledge acquisition process during the early design phase [ROJ_11a].

2.3. Structure, simulation process, and modularity of MOVES^2

The structure of the MOVES^2 is represented on Figure 2.2. Three expandable libraries store the information used during the simulation process. The *Functions library* collects all those general functions that cover specific user needs, such as, to run a simulation, to identify unknown system parameters based on measured data (e.g. the engine mass, the car body inertia in specific a direction) or to load an animation. These functions normally use the content from the other two libraries and eventually another general function. The *Models library* contains the Simulink models and the *Parameters library* stores the information required to execute simulations regardless the used model. This last library collects the parameter scripts. A parameter script is a file that groups known parameters from the same field. An example of a parameter script file is the road alignment parameter file, which collects the minimum amount of parameters to define the road profile in a macroscopic level.

The information can be executed, changed, modified or selected using specific parts of the GUI of MOVES^2, or directly through the MATLAB command window and the MATLAB editor.

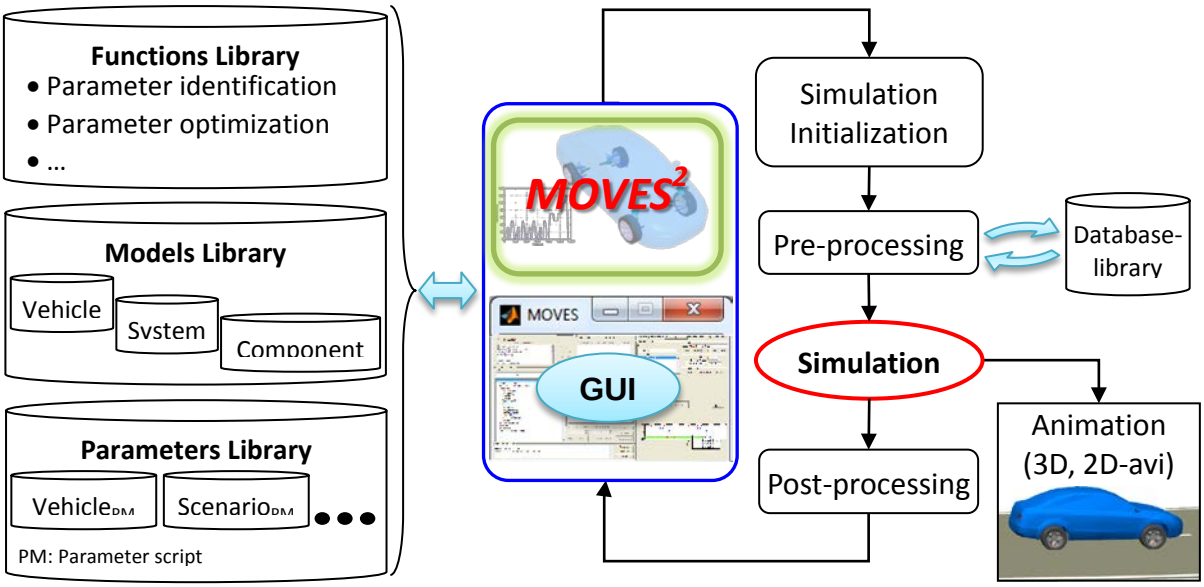


Figure 2.2. Structure of the MOVES^2 and simulation process carried out in it [ROJ_11a].

The standardized programming methodology defines how to manage the information during the simulation process. In the *Simulation initialization* the selected parameter script files are arranged in a structured way. During the second step a *pre-processing process* calculates variables based on parameters from the structure, saving them in the structure as well. Moreover, this process occasionally uses files from a database-library. If this is the case, a parameter from a parameter script contains the name of the required database-library file. The database-libraries group data from specific fields, such as customized trajectories or speed profiles. The data from this type of files is usually arranged by tables since they may contain heavy data. This *pre-processing process* is entirely composed by those functions specified in the parameter scripts. Then the simulation is automatically run, and the animation can be watched parallel to the simulation. Afterwards, the *post-processing process* organizes the simulation results within the structure and executes post-processing functions (e.g. to move all the necessary files to play the animation into the simulation results folder). Finally, if the simulation has been executed using the GUI, the structure of variables is displayed on it, in order to be visualized, edited or saved by the user.

To use MOVES^2 during the entirely early design phase requires different models or a flexible one with an adaptable complexity. The Simulink model from MOVES^2, developed at the Institute of Automotive Engineering at Graz University of Technology, can adjust its modelling complexity organized in three model depths: vehicle, system and component level. Thanks to the linkages of the *Models library* with the model implemented in MOVES^2, the specification of its systems and components can be changed. These linkages are based on configurable subsystems: the *block choices*.

Figure 2.3 shows examples of different modelling complexity, from the model integrated in MOVES^2, at different model depths. According to the vehicle level, the model can be configured as a quarter, single track or full vehicle model. The number of degrees of freedom – DOF – corresponds to two, from three to seven and it is increased by adding suspended bodies to the chassis, respectively. The

figure illustrates an example of system and of component: the steering system model and the tire model, respectively. The model can specify a sort of steering system models and tire models with different modelling complexities [ROJ_11a].

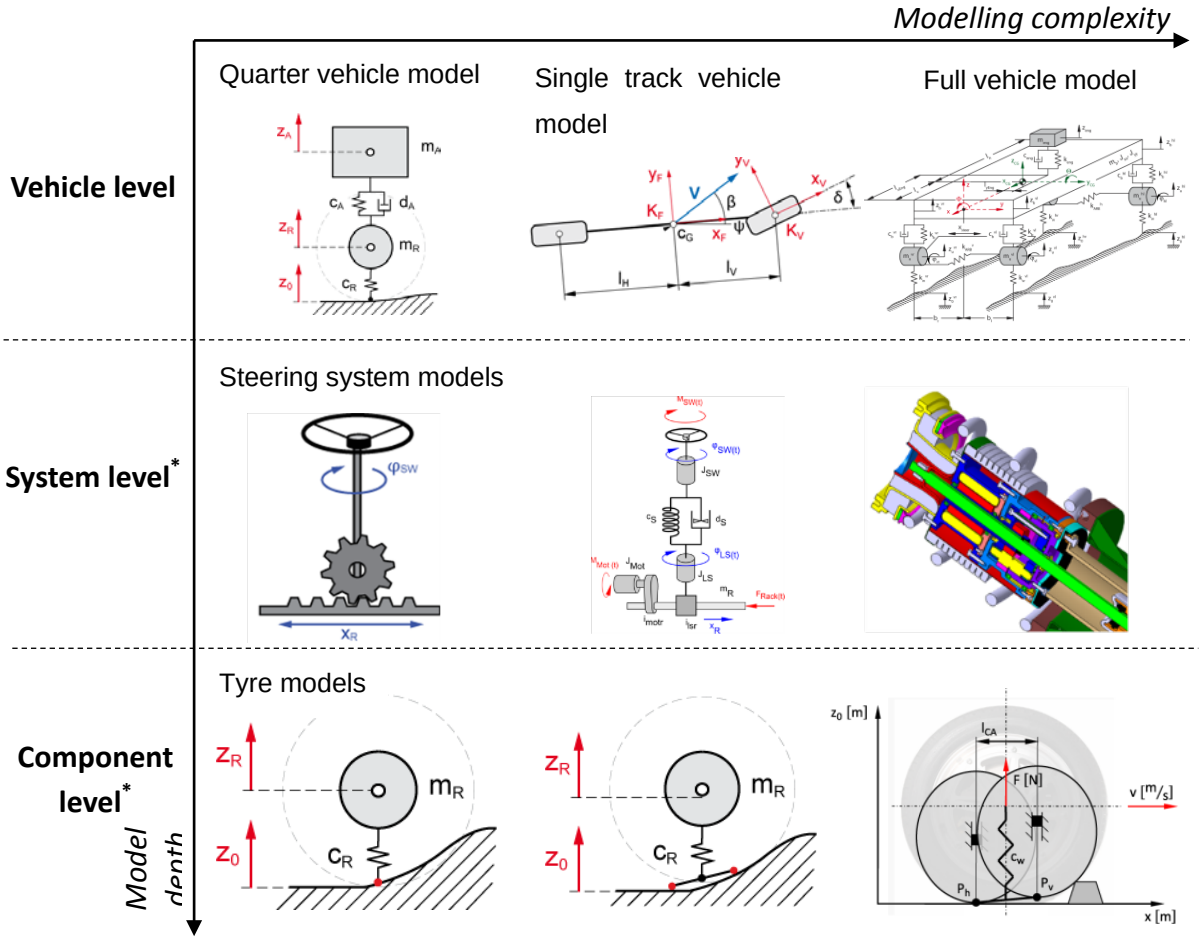


Figure 2.3. Model modularity in the MOVES[2] [ROJ_11a].

2.4. Driver-Vehicle-Environment system

Before starting a simulation, the simulation process requires the vehicle, environment and driver controller totally defined (see Figure 2.4). Parameter script files define the system with several optional characteristics available (e.g. to add control systems, to set obstacles).

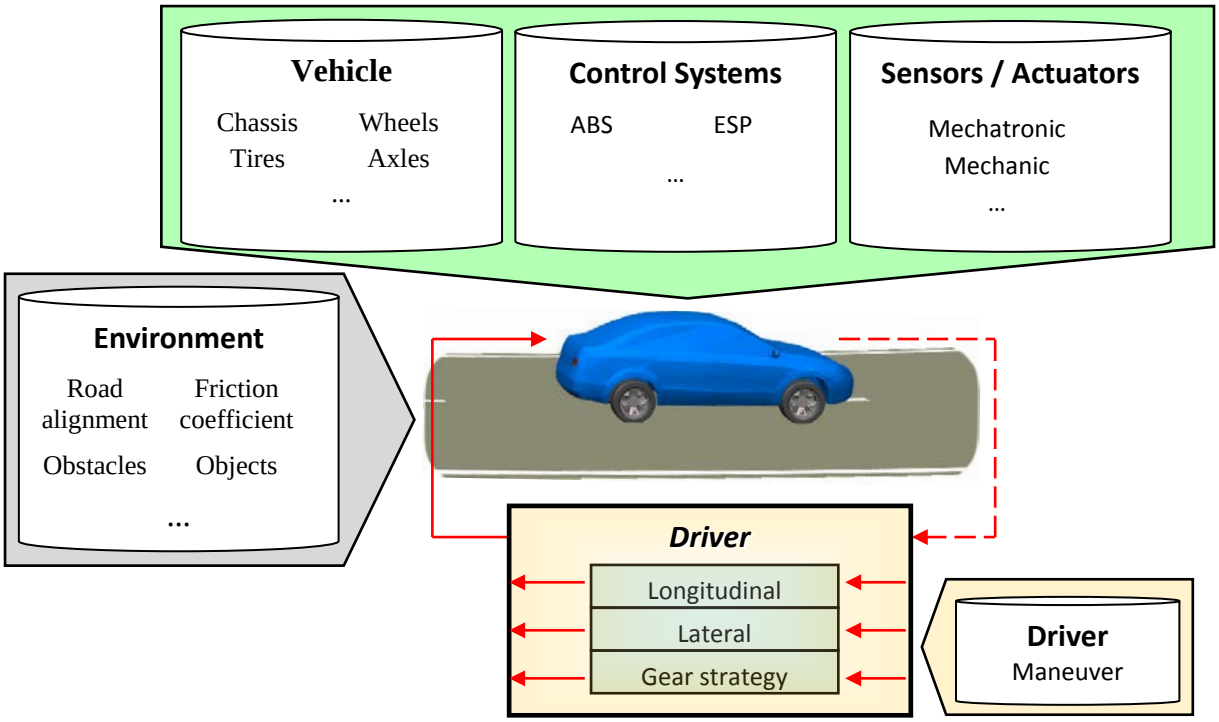


Figure 2.4. Driver-Vehicle-Environment system [ROJ_11a].

The vehicle definition groups the vehicle parameters itself, the control system options and the sensors and actuators, while the environment collects the road geometry definition with the road perturbations (surface roughness and obstacles such as bumps or holes), the objects selection (e.g. cones, balls, trees), and the road friction coefficient definition. The driver controller is responsible for the longitudinal and lateral vehicle motion, and for the automatic or manual gearbox. Furthermore, standardized open-loop and close-loop manoeuvres as well as customized manoeuvres created with the manoeuvre-building module can be selected [ROJ_11a]. The standardized double lane-change manoeuvre (known as severe lane-change also) is a common close-loop manoeuvre to evaluate the lateral stability of a vehicle. It consists of a left lane-change followed by a right lane-change without exceeding lane boundaries and performed over 125 meters on a straight road. Cones mark the driven area of the road. This manoeuvre is specified in the international standard ISO 3888-1 [ISO_99].

2.5. Manoeuvre-building module

The manoeuvre-building module developed in this work allows the creation of trajectories in a three different ways:

- The first option is to use the road database-library. In this case the target trajectory follows the midline from a previously created road. In addition, this midline may be shifted to the left or right hand side according to the current road set in the environment. Figure 2.5 illustrates a circuit where the target trajectory has taken from the road set in the environment (the circuit) and its midline (*Dashed line*) is shifted 5,5 meters to the right hand side.

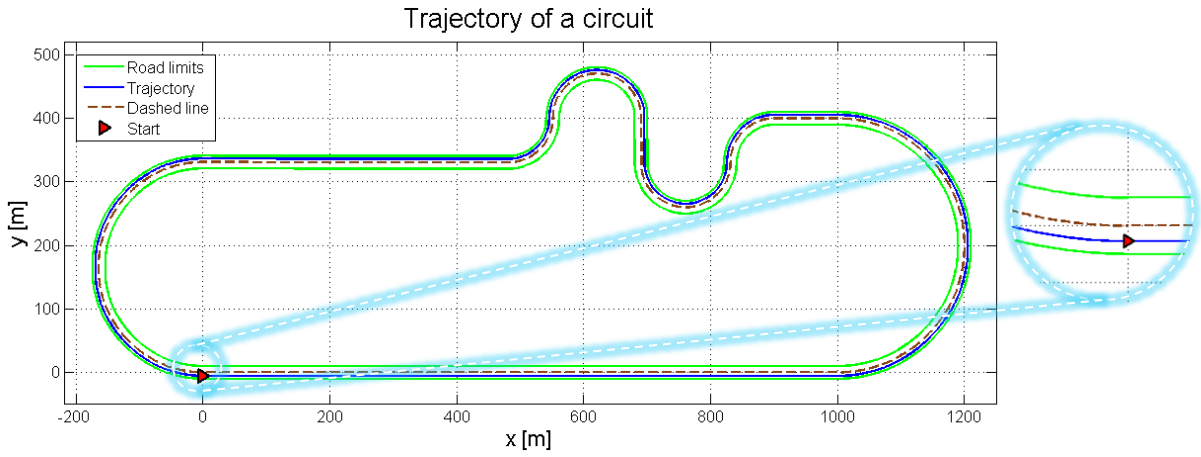


Figure 2.5. Customised trajectory using the first option of the manoeuvre-building module: road database-library. Trajectory of a circuit shifted 5,5 meters to the right.

- The second option is to use the trajectory database-library to select new customized trajectories. Trajectory files can set the target trajectory after being aligned or shifted to the right or left from the midline of the current road. It is noteworthy that the Simulink block *Signal Builder* of MATLAB/Simulink® has been used in the GUI of the MOVES[2] to facilitate the creation of new user-defined trajectories. This block can create and generate $x-y$ signals with discretized waveform [MAT_11]. Figure 2.6 exemplifies the double lane-change manoeuvre test. In this case, the target trajectory is shifted 1,7 meters to the right hand side from the road midline.

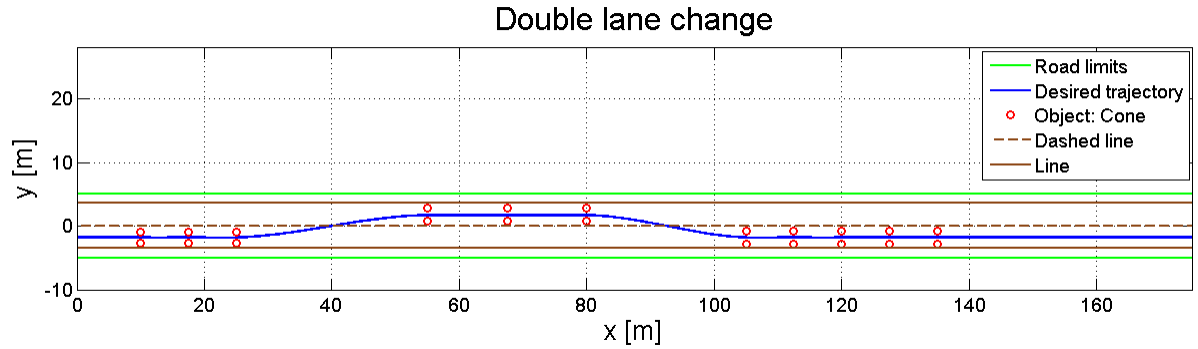


Figure 2.6. Customised trajectory using the second option of the manoeuvre-building module: trajectory database-library. Trajectory of the standardized double lane-change [ISO_99] generated with the Signal Builder block.

- The third option is the use of trajectory functions. These user-defined functions have a high level of flexibility. They create the $x-y$ coordinates of the desired trajectory based on the information of their input variables. Figure 2.7 shows a sinusoidal trajectory on a straight road. In the figure, the user-defined function set the trajectory as a sinus with amplitude of 10 meters, period of 50 meters and initial phase of $-\pi/2$ radians.

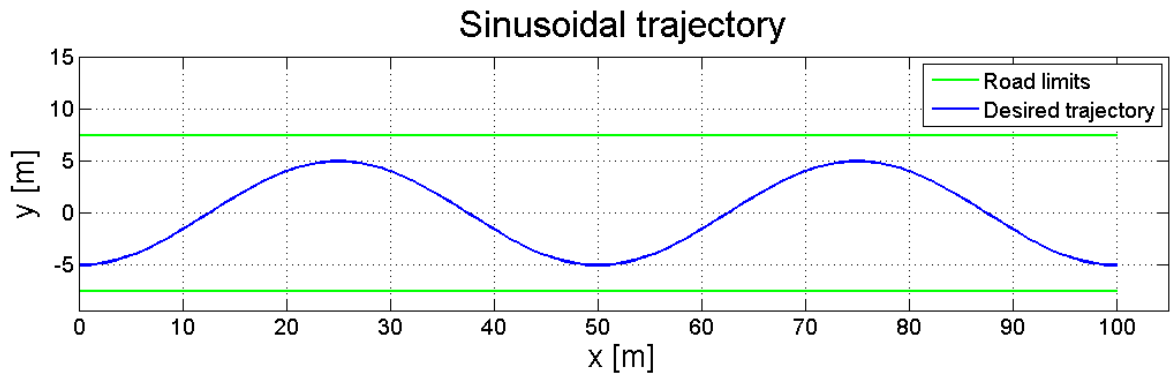


Figure 2.7. Customised trajectory using the third option of the manoeuvre-building module: trajectory functions. Sinusoidal trajectory with 10 m. of amplitude, 50 m. of period and –π/2 rad. of initial phase.

The manoeuvre-building module creates trajectories in *Cartesian coordinates*, and afterwards the *heading angle* and *driven distance* values are calculated, since the driver controller – implemented in MOVES² – requires them. Finally the trajectory can be saved. When no option to create the target trajectory is specified, the manoeuvre-building module uses as the default target trajectory the road midline.

Nevertheless, the real trajectory of a vehicle is the result of the whole driver-vehicle-environment system specified in the simulation process. Therefore, target and real trajectories do not necessarily coincide. The following figure illustrates the target and the real trajectories for the standardized double lane-change test, where the entry and exit speed from track test are both 18 m/s.

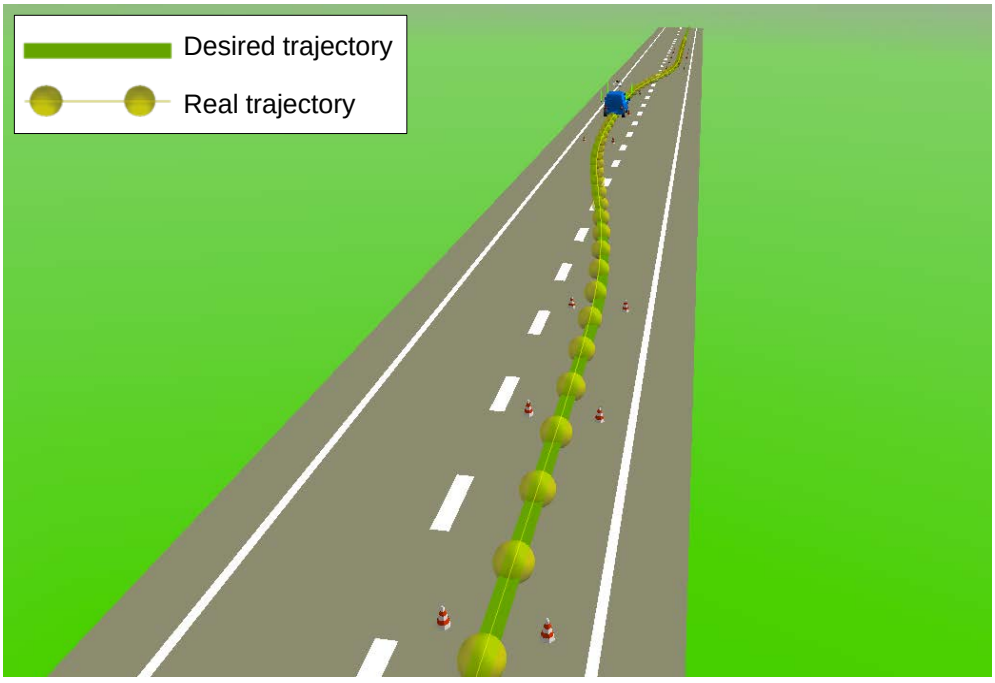


Figure 2.8. Desired and real trajectories in a severe double-lane change test [ISO_99] driven at 18 m/s. Screenshot of the visualization feature implemented in MOVES² as part of the present work.

3. THE THREE-DIMENSIONAL ROAD

This chapter presents the three-dimensional road used in the full vehicle model of MOVES[2]. However, two-dimensional vehicle models, such as the quarter and the single track vehicle model of MOVES[2], may use the three-dimensional road by determining the wheel contacts for a specified trajectory.

The road-builder generates road specifications, which define the three-dimensional roads. They group the road profile and the road friction coefficient definition.

Within the road profile, the road geometry, viewed from a macroscopic standpoint (straights and curves), together with the road perturbations (e.g. holes and bumps) and road roughness (according to the International Standard ISO 8608 [ISO_95]) are defined. The obstacle-builder, developed in this work, is presented in this chapter. It allows the addition of user-defined road perturbations.

The coefficient of friction is an optional specification. If it is not defined, a default friction coefficient of one ($\mu_{default} = 1{,}0$) is applied for the simulation.

Several companies have developed tools to generate roads, due to its importance, in a three-dimensional environment. MOVES[2] integrates the open source OpenCRG® program as a basic tool to generate roads [VIR_10].

To cope with realistic road designs when the road-builder is used, this chapter also provides an overview of the general aspects in the geometric design of roads.

3.1. Coordinate systems

In a road-vehicle simulation, the road is described as a grid of points. The position of these points is usually described in a u-v coordinate system. The u coordinate gives the position on a reference road longitudinal line (commonly the midline) and v coordinate describes the lateral position of the from this reference line [ATK_02] [MA_95]. This procedure is a parameterization of the road surface and the u-v coordinates are called *surface coordinates*.

Definition. A parameterized space curve is a class C^1 mapping[1] with

$$\begin{aligned} \sigma : I &\longrightarrow \mathbb{R}^3 \\ u &\longmapsto \sigma(u) = (x(u), y(u), z(u)) \end{aligned} \qquad \textit{(Eq. 3.1)}$$

where σ is defined on a closed interval of real numbers $I = [a, b]$ of $\mathbb{R}$, and the u variable is called *parameter* [PAS_05] [SHI_95].

[1] Class C^1 denotes a function that is continuously differentiable in its domain, and its derivative is continuous [GIA_04].

Definition. A parameterized surface is a class C^1 mapping with

$$\varphi : D \longrightarrow \mathbb{R}^3$$
$$(u,v) \longmapsto \varphi(u,v) = (x(u,v), y(u,v), z(u,v)) \qquad \textit{(Eq. 3.2)}$$

where φ is defined on domain D of $\mathbb{R}^2$, and φ is injective[2] in D, and $S = \varphi(D)$ is called a *parametrically defined surface*. The u,v variables are called *surface parameters* or *surface coordinates* [PAS_05] [SHI_95].

The road parameterization combines its road reference line parameterization with the usage of a second variable to define the entire surface. Then, the u variable is both the *parameter* of the parameterized space curve (reference line) and a *surface parameter* of the parameterized surface (road), and the v variable is the second *surface parameter* of the parameterized surface. In other words, u defines the road reference line and v extends the mapping to cover road points from given reference line positions. How the v variable locates road positions out of its reference line is explained by how this *surface coordinate* is defined: let S be a parameterized road surface by the u,v *surface parameters*, the u and v vectors, described in *Cartesian coordinates*, are orthogonal at any point along the reference line. Thus the geometry of the reference line defines the orientation of these vectors.

The following figure shows a vehicle driving on a zigzag road of 48 meters in length and 7 meters in width. It illustrates the orientation of the u-v coordinate system for a given vehicle position. Furthermore, this position is indicated in both coordinate systems. In this example, the vehicle is currently at $x =$ 26 meters and $y =$ -3 meters or at 28,6 meters and 2 meters in u-v *surface coordinates*.

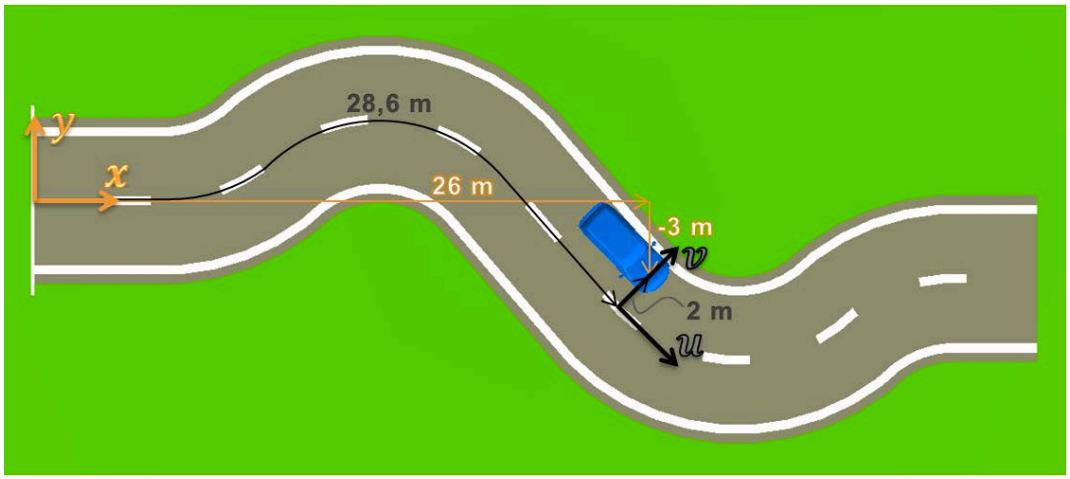


Figure 3.1. Vehicle position measured in x-y and u-v coordinates. Screenshot of the visualization feature implemented in this work.

[2] A function f with its domain D, is injective – commonly known as *one-to-one function* – if only if $\forall\, a,b \in D$, if $a \neq b$, then $f(a) \neq f(b)$ [PAS_05].

As it can be noticed, the *parametrically defined surface* deals with the *surface coordinates* instead of the *Cartesian coordinates*. As a consequence, calculations regarding the road are usually faster (e.g. to set obstacles). Anyway, the *rectangular coordinates* are sometimes more useful, e.g. to calculate the *driven distance*. Therefore it is necessary to go back and forth between these two coordinate systems.

Because of the need to use discrete road surfaces as grid of nodes, *surface coordinates* have the following requirements:

$$\begin{cases} u_{\mathrm{i}} = \lambda\,, & i = 1 \\ u_i = u_1 + \sum_{k=2}^{Floor(\alpha \cdot i)} \left\| \left[\begin{pmatrix} x_k \\ y_k \end{pmatrix} - \begin{pmatrix} x_{k-1} \\ y_{k-1} \end{pmatrix} \right] \right\|_{ref\ line} \Bigg\|\,, & i > 1 \end{cases} \qquad \textit{(Eq. 3.3)}$$

$$v_j \in \left[-\frac{width_i}{2}, \frac{width_i}{2} \right], \quad \forall j \in D \qquad \textit{(Eq. 3.4)}$$

where λ is the origin of u coordinate, $\begin{pmatrix} x_k \\ y_k \end{pmatrix}\Big|_{ref\ line}$ corresponds to the reference line in *Cartesian coordinates* of the road (indexed by k), $Floor(x)$ is the function that gives the integer part of x, α is the relation between the reference line discretization (the distance from one point to the following one) and the u discretization – can be calculated as Δx divided by Δu. D is the domain of the parameterized surface, and $width_i$ is the road width at u_i – calculated as the perpendicular distance (from one side of the road to the other side) to the reference line through u_i.

3.2. Geometric road design

A road network must be designed to ensure safety to pedestrians and vehicles. Criteria to guide road geometry design are based on road types, design vehicle speed, view obstructions, sight distances, and the terrain where it is built. Governmental entities of Transportation from different countries (such as Department of Transport in UK, National Swedish Road Administration in Sweden, etc.) have created standards to provide guidelines in order to design roads. Despite of this plurality of standards, main criteria are shared by all countries and just the singularities are different between countries (e.g. estimate coefficients, specific formulae) [AAS_04] [FIT_07] [WAK_98].

Concerning the macroscopic aspect, the road profile is described by the superposition of its horizontal and vertical alignment. The horizontal alignment of a road determines its succession of *straights* and *horizontal curves* – a *horizontal curve* may have *superelevation* (known as *banking*) but not *grade* (known as *slope*) inclination. I.e. horizontal alignment is the road geometry seen from a helicopter. Alternatively, the vertical alignment defines the altitude of a road.

3.2.1. Horizontal alignment

Horizontal alignment belongs to the x-y plane, and it consists of a succession of *tangents* (*straights*) and constant radius *curves* that may be connected by *transition curves* with non-constant radius called *clothoids*. Furthermore, horizontal alignment also groups what is called *superelevation* (known as *banking*) that aids in steering of vehicles on curves. According to standards from governmental entities of Transportation from different countries, the radius of a curve is defined by its *design speed*:

$$R = \frac{v^2}{\mathrm{C} \cdot (e + f)} \qquad \textit{(Eq. 3.5)}$$

where R is the curve radius [m], v is the *design speed* [km/h], C is a conversion constant (equal to 127 for metric units), e is the curve *superelevation* [-], and f is the available *side friction factor* [-] which is the friction percentage used for turn the vehicle and it depends on the vehicle speed. Its value differs among countries standards [FIT_07] [MIN_00].

Because of the radius of *transition curves* changes linearly with the curve, they improve comfort and safety when vehicles negotiate curves. Thus, *transition curves* are *clothoids* (*Euler spiral*) with the following equation:

$$R \cdot L = A^2 \qquad \textit{(Eq. 3.6)}$$

where R is the curve radius at any point P on the spiral [m], L is the length of the spiral from its *point of inflection* (where $R = \infty$) to the point P [m], and A is the clothoid parameter [m] [MIN_00].

The length of the spiral should accomplish requirements related with the variation of the lateral acceleration, and visual perception factors [MIN_00]. Although, in road construction, it is common to establish the length of the *transition curve* between a third of the curve radius and the curve radius [VIR_10]:

$$\frac{R_{curve}}{3} \leq L \leq R_{curve} \qquad \textit{(Eq. 3.7)}$$

with R_{curve} as the radius of the curve linked by the *transition curve* [m], and L as the length of the *transition curve* [m].

To easily work, mathematics defines the *curvature* variable of a curve as:

$$\kappa = \frac{1}{R} \qquad \textit{(Eq. 3.8)}$$

where, κ is the *curvature* of the curve at point P [m^{-1}], and R is the radius of the curve at P [m] [CAS_96].

In general, *curvature* values can be positive, negative, and zero values along a curve. When a point P has κ equal to zero it means P belongs to a straight line, when κ is positive P belongs to a left turn, and if P has a negative κ, P belongs to a right turn [CAS_96].

Figure 3.2.a illustrates a trajectory with a *transition curve* overlaid with the corresponding Euler spiral. Figure 3.2.b shows its *curvature* as a function of the *driven distance* and it illustrates how κ increases linearly along the curve. In this example, the *transition curve* starts, after 3 meters *tangent*, to increase its *curvature* from zero up to 0,2 m^{-1} along 4 meters length. Afterwards, this trajectory turns left during 7,5 meters following a 5 meters radius curve. *Curvature* values have been calculated with the 3 points circle placing method [KRA_11].

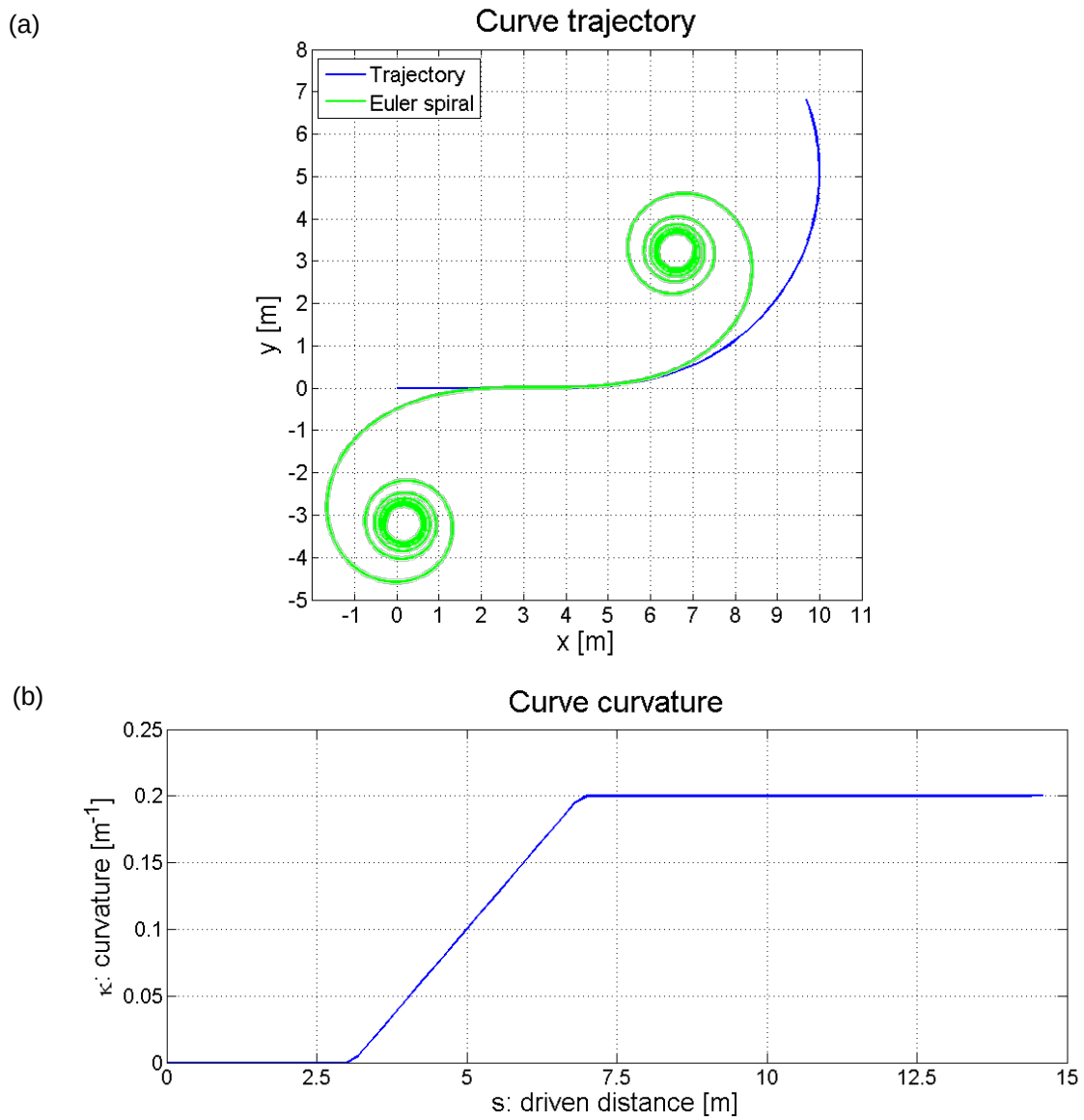


Figure 3.2. Trajectory composed by a tangent followed by a curve with transition, with the superposition of its Euler spiral (a) and the driven distance of the trajectory vs. its curvature (b).

Eq. 3.5 establishes a relation between *superelevation* and the available *side friction factor* for a fixed *design speed*. Hence, curves may have *superelevation* factors. Many country's standards recommend maximum *superelevation rates* of 6 to 8 per cent, but Australian standards define the maximum rate depending on the terrain type. For flat terrain it is defined as 6 per cent, although for mountainous terrain the maximum *superelevation rate* goes up to 12 per cent. German standards define the maximum *banking rate* as 8 per cent.

Furthermore, USA standards suggest five methods to calculate and set the *superelevation* distribution (see Figure 3.3). Every method assumes a specific relation (e.g. direct proportionality, successive addition) between *superelevation* and *side friction factor* of the road [FIT_07].

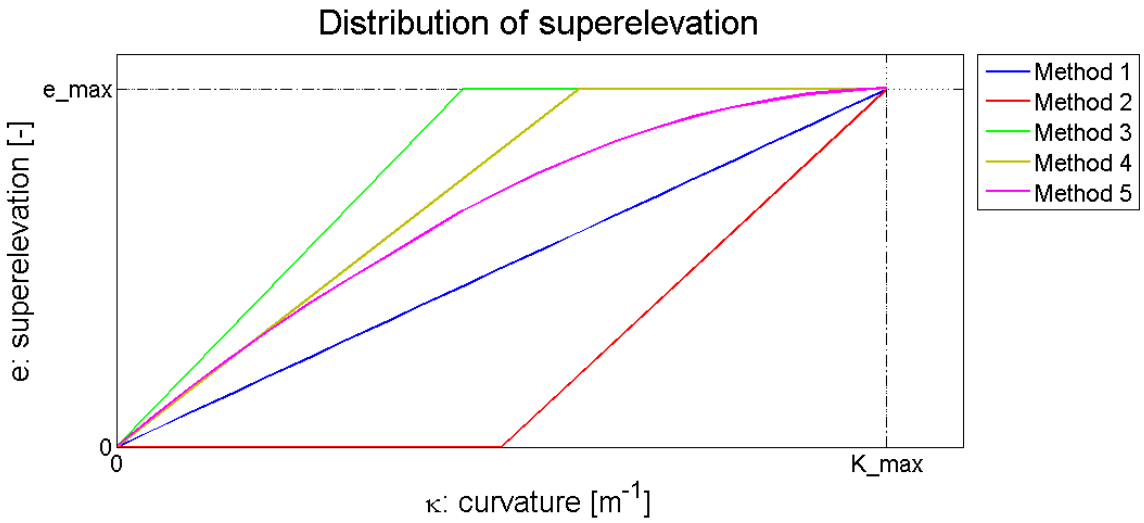


Figure 3.3. Methods to distribute banking along the transition curve [FIT_07].

Method 1 assumes *superelevation* and *side friction factor* developed with a direct proportionality with *curvature*. Method 2 assumes that first is used the available *side friction factor*, and once the maximum *side friction* is reached, $banking$ is increased to achieve higher speeds, which is just the opposite of methods 3 and 4. Method 5 is the result of an algorithm that first uses more *side friction* than *superelevation*, and then it uses more *superelevation* than *side friction* at large radii. It is noteworthy that for high-speed roadways method 5 is used in the USA, and in Spain – as in many countries –is generally used method 1 for any type of road. Finally, it is important to note that the transition from the normal road profile to the banked profile is distributed in the *transition curve* [FIT_07] [MIN_00].

3.2.2. Vertical alignment

Vertical alignment belongs to the vertical plane and it is compost of vertical curves: two tangents connected with one parabolic arc. Vertical curves are sorted into two types – *sag curves* and *crest*

curves – and both of them modify *slope* (commonly known as *grade*) up to 8~10% with a constant rate of change. The difference of *slope* percentage before and after a vertical curve defines its type. *Sag curves* increase *grade* and *crest curves* decrease it, i.e. the road that crosses a valley may contain *sag curves*, and to cross a hill a *crest curve* may be used. It is extremely important use the correct set of vertical curves (the profile grade) to guarantee safety and good drainage over the roadway [FIT_07] [MIN_00].

Figure 3.4 illustrates two vehicles travelling on a crest vertical curve with 10% and -10% of *grade* maximum rate. In general and as the picture shows, *crest curves* may have visibility problems, as well as for *sag curves* during the night. Hence, special criteria limits *crest* and *sag* curves. On the one hand, the limitation of *crest curves* is the sight distance, the distance ahead the road user where he has visibility of stationary or moving objects on the road. On the other hand, headlight sight distance restricts *sag curves*, while over the night car lights light different distances for different vertical alignments [AAS_04].

Figure 3.4. Road profile with a crest vertical curve with 10% of maximum slope rate. Screenshot of the visualization feature implemented in this work.

3.3. OpenCRG®

To generate roads, the MOVES[2] integrates the open source OpenCRG® [VIR_10]. Nevertheless other tools to generate roads are currently available (e.g. cosin/3Droad [GIP_10] or RoadXML© [CHA_10]).

The OpenCRG® initiative started in October 2008, it was funded by the automotive OEMs[3] Audi, BMW, Daimler, Porsche, and Volkswagen, and it has been developed by VIRES Simulationstechnologie GmbH. The objective of the project is to create and provide an open file format and tool-suite – developed in MATLAB® – for the creation, management and evaluation of high precision 3-D road surface descriptions with a compact storage. The first release of OpenCRG© was published on April 2010 [DAI_10] [VIR_11].

[3] OEM: [abbreviation] Original Equipment Manufacturer (an organization that makes devices from component parts bought from other organizations) [NOA_10].

The name OpenCRG® is based on two key aspects: to be an open source, and to use the so-called Curved Regular Grid – CRG – roads. The basic idea of the CRG method is to parameterize the road surface with the *surface coordinates* u,v as a rectangular grid. Thereby, u coordinate defines the Curved Reference Line – CRL –, usually the midline of the road (see Figure 3.5), which sets the road direction (*heading angle*) and optionally its *slope*. The rest of the road nodes are arranged in u direction with offset lines from the CRL. The v coordinate locates the road nodes in the direction which is perpendicular to the CRL. Based on this definition, additional cross slopes can be defined (e.g. *superelevation*) [DAI_10].

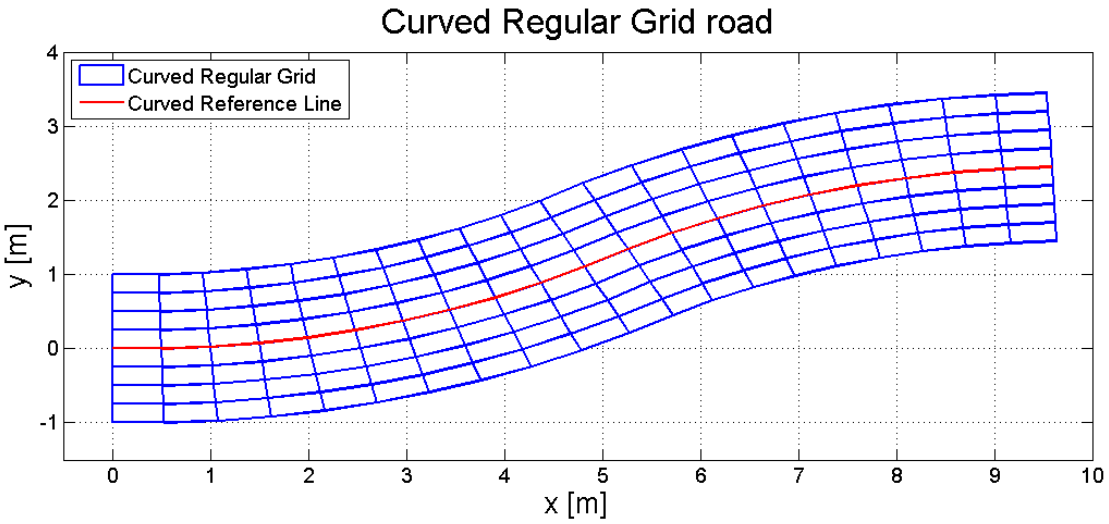


Figure 3.5. A Curved Regular Grid road description with its Curved Reference Line – CRL – highlighted.

The road discretization may be different for different simulation purposes. As a consequence, OpenCRG© discretizes roads with predefined gaps in nodes for both u and v directions. In other words, increments in u direction and increments in v direction locate the road nodes.

Figure 3.6 illustrates those increments and shows the geometrical requirements in Cartesian coordinates of the nodes that are aligned through offset lines to the Curved Reference Line. As it can be seen from the figure, the gap in nodes from the CRL (midline road) is constant, but not the gap in nodes for v direction, neither the gap in nodes for u direction out of the reference line. The constant increment in v direction is performed by the offset lines to the CRL, with Δv as offset.

As it has been explained in section 3.1, *surface coordinates* are orthogonal, thus if u direction is known, v direction should be easily determinable. However, to trace perpendicular lines in a discrete space is not trivial, so geometric concepts may be used. In this case, an isosceles triangle can get the perpendicular line – to the curve that is build up by two straight segments of the same length – if the triangle base is out of the CRL. Then the locally perpendicular line to the reference line is the same line as the perpendicular one to the triangle base through its opposite edge [VIR_11].

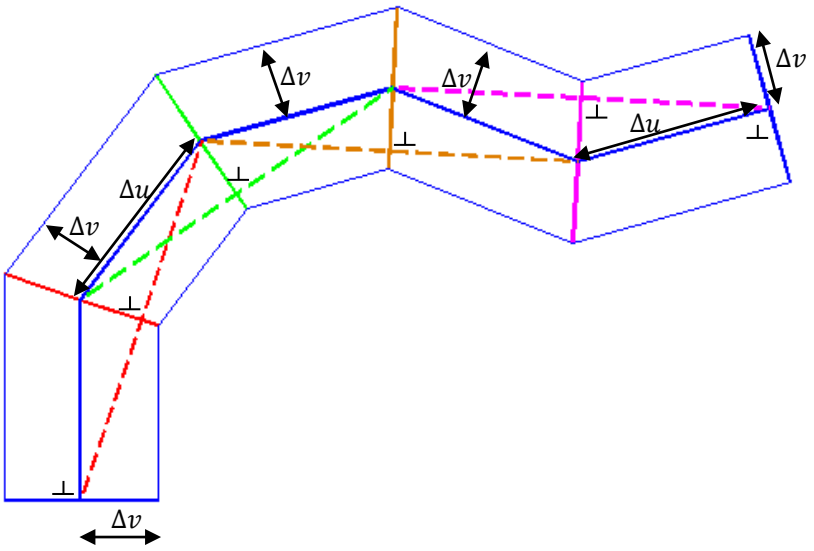


Figure 3.6. Geometrical requirements of offset lines from the Curved Reference Line in a Curved Regular Grid [DAI_10] [VIR_11].

When OpenCRG© is run in MATLAB®, the road information is organized in a specific structure. The CRG data structure consists in different fields that define the road in a compact way, thus, according to its shape, the road definition requires certain number of variables with a specific size. Briefly, the relevant structure fields are explained below.

First, a head field contains the structure with important data scalars regarding the reference line (e.g. the u,v and x,y origin and end of the reference line, the distances between road nodes, the start and end value of *heading angle*, *slope*, *elevation* and *banking* of the reference line), and another scalar field defines the *length* the road. Then, on the one hand, vector fields establish the horizontal alignment of the road. The information of *heading angle* field defines this property into the road midline, but *banking* field gives the *superelevation* to the nodes with the following method: first *banking*'s number sets first line of nodes arranged in v direction, second *banking*'s number sets second line of nodes in v direction, and so on. On the other hand, *slope* vector field defines the vertical alignment corresponding to the road midline. Moreover, other fields have information related to CRG modifiers and processing options [VIR_10].

3.3.1. The CRG-road, main features

OpenCRG® contains a tool-suite that provides three different kinds of features: the CRG data generation, the modifiers and options, and the functionalities.

3.3.1.1. CRG data generation

With the suitable information, the tool-suite of OpenCRG® can generate Curved Reference Lines and Curved Regular Grids. The Curved Reference Line – known as reference line – is a partial CRG data

structure, containing information just to describe the reference line (see Figure 3.7). Despite of that fact, it is easy to create CRLs from measured data because it requires just a vector of x-y points.

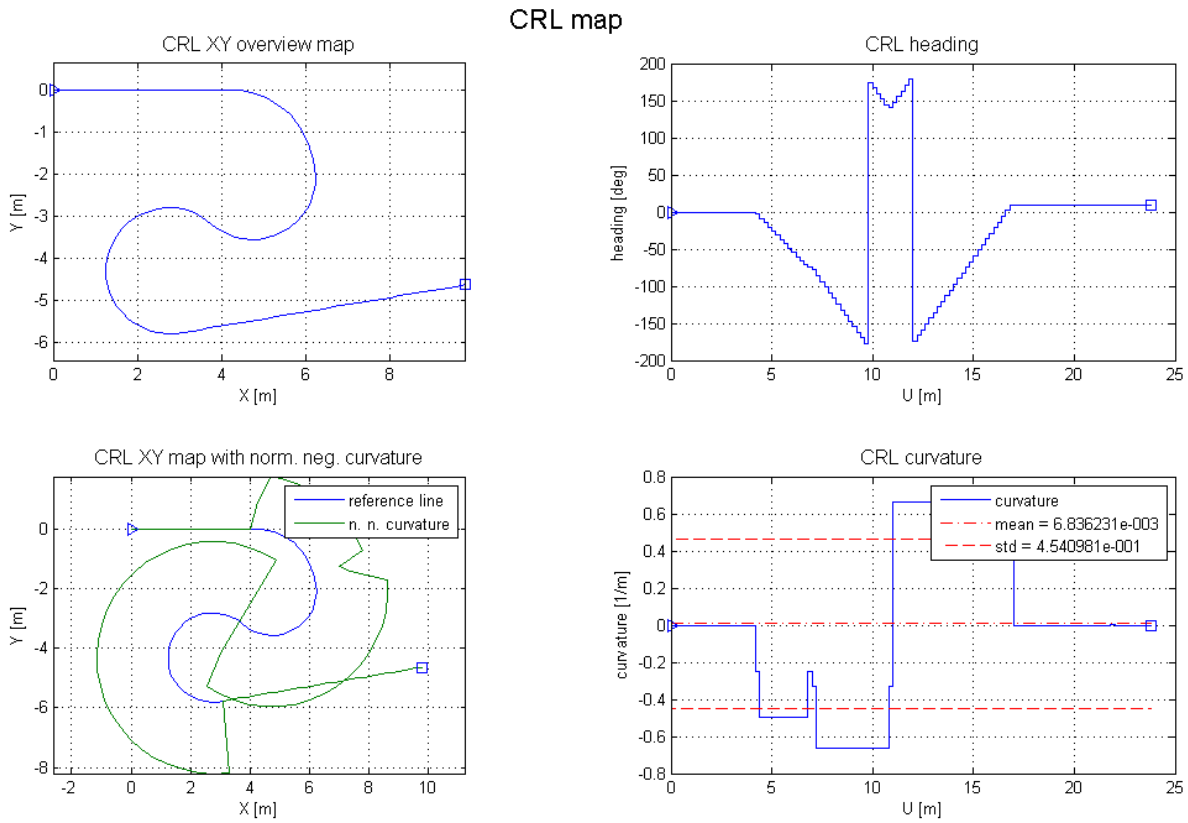


Figure 3.7. Plots from the information of a Curved Reference Line.

Curved Regular Grid – known as road – makes use of the entire CRG data structure. The procedure to generate CRGs needs global information about the road (e.g. *length*), and it uses specific information to define vertical and horizontal curves. The minimal input parameters required to create a road are the ones grouped by the global characteristics. Those parameters are the *road length* and *road width* as well as the *gap between nodes* concerning the u and v directions. Figure 3.8 illustrates a road with the minimal information described in *Cartesian coordinates*. In this case, the road has the length of 7m with a width of 2m and node increment of 0,5m in both directions.

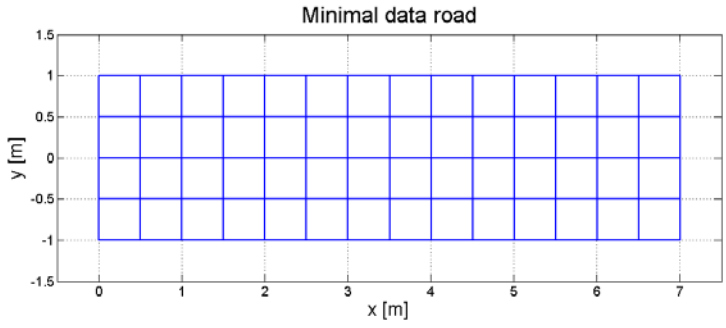


Figure 3.8. Road described in Cartesian coordinates with the minimal information to be generated.

Defined sections of the road are characterized by its *curvature*, *banking* and *slope*. Each characteristic is treated separately and hence it is possible to generate any road geometry.

The *curvature* is described by a polynomial of degree up to 1 for every curvature section (independent of banking or slope sections) with the following coefficients:

$$\begin{aligned} c_0^i &= \kappa_s^i && ,\forall i \\ c_1^i &= \frac{\kappa_e^i - \kappa_s^i}{LC^i} && ,\forall i \end{aligned} \qquad \textit{(Eq. 3.9)}$$

where c_0^i and c_1^i are the constant term and the first polynomial coefficient respectively for the i curvature section length (LC^i) [m], and κ_s^i and κ_e^i are the i section starting and ending *curvature* respectively [m^{-1}] [VIR_10].

Equivalently, *banking* (or *superelevation*) is described by a polynomial of degree up to 1 for every banking section. Sections are independent of the ones defined in *curvature* and *slope*. The following coefficients characterize each banking section:

$$\begin{aligned} b_0^i &= e_s^i && ,\forall i \\ b_1^i &= \frac{e_e^i - e_s^i}{LB^i} && ,\forall i \end{aligned} \qquad \textit{(Eq. 3.10)}$$

with b_0^i and b_1^i as the constant term and the first polynomial coefficient respectively for the i banking section length (LB^i) [m], and e_s^i and e_e^i as the i section starting and ending *banking rate* respectively [-] [VIR_10].

Finally, the vertical alignment is defined through a polynomial of degree up to 1 as well. The *slope* (or *grade*) definition makes use of different slope road sections (independent of curvature or banking sections) with different coefficients (Eq. 3.11).

$$\begin{aligned} s_0^i &= \delta_s^i && ,\forall i \\ s_1^i &= \frac{\delta_e^i - \delta_s^i}{LS^i} && ,\forall i \end{aligned} \qquad \textit{(Eq. 3.11)}$$

where s_0^i and s_1^i are the constant term and the first polynomial coefficient respectively for the i slope section length (LS^i) [m], and δ_s^i and δ_e^i are the i section starting and ending *slope rate* respectively [-] [VIR_10].

Curvature, banking and slope polynomials can be up to degree 1, depending on the number of coefficients. If only its first coefficient is different than zero, this polynomial has grade 1 and its performance keeps the parameter value defined in it (*curvature*, *banking* or *slope*). Otherwise, if both coefficients are not null, the corresponding polynomial has degree 1 and its performance causes a linear increment of the parameter value of the polynomial. Finally, if no polynomial is defined within a road, the property of the polynomial (*curvature*, *banking* or *slope*) will be null for this road.

Consequently, the road generation of OpenCRG® is totally flexible; curvature, banking and slope polynomials describe roads with any horizontal and vertical alignment.

3.3.1.2. Modifiers and options

The CRG data structure is generated or loaded to accomplish some needs or functionality. However, this requirement not always uses the entire data. For example, it may need just to evaluate a road, or it may need beforehand a scaling of the road elevation.

With modifiers and options it is possible to first modify the saved CRG data to reach evaluation requests, and second to change specific performances of CRG data without modifying the file. The information regarding both concepts is stored in two structure fields inside the CRG data structure. The information from the modifiers field can perform the following purposes: to scale elevation data, to relocate the reference line with an $x\ y\ z$ or *heading angle* offset, and to handle the undefined elevation values of the node grid as zero, as undefined value, or keeping the last value in cross section. The main characteristics from the options field are: to set the border nodes either in v or u direction as undefined value, as zero, keeping the last value, assigning the value from the opposite one, or repeating the border value, to define the reference line as open or close track, and to enable CRG message options [VIR_10].

3.3.1.3. Functionalities

The tool-suite of OpenCRG® collects a set of functions in MATLAB® to evaluate, modify, analyse, and visualize the CRG data.

The evaluation tools allow the evaluation of *curvature* or *heading angle* at reference line u positions, the evaluation of z elevations at x-y or u-v grid positions and the transformation from the x-y to the u-v coordinate system and vice versa. Moreover, this set of tools can work with WGS 84 positions. The World Geodetic System – WGS – is the standard coordinate system for the Earth, and specifically, the standard from 1984 with revisions from 2004 – WGS 84 – is the currently used system. Then, it is possible to transform points from the x-y system to WGS 84 coordinates, to evaluate *distance* and *bearing* (or *azimuth* direction) from WGS 84 positions, to calculate WGS 84 position by *distance* and *bearing*, to generate html files to show the track in Google Maps (© Google), and to show WGS 84 coordinates on URL[4] string.

Modification tools can mainly filter, limit, map, and re-render CRG data, add and extract $banking$ and $slope$ from the *elevation* data and cut and merge CRG roads.

Analysis tools can perform the comparison of two CRG data structures giving as result the Boolean in order to identify the road identically, and a structure array with further comparison information (e.g. u

[4] URL: [abbreviation] uniform (or universal) resource locator, the address of a World Wide Web page [NOA_10].

and v space). The other tool is to find peaks within a CRG road. In this case, the detection is indexed with the peak index variable and the u-v peak position.

Visualization tools show geometry data of the reference line and the road surface, with 2-D and 3-D plots. E.g. to plot the *curvature*, *heading angle* or *slope elevation* of the reference line, to plot the whole road in the x-y-z or u-v-z coordinate system, to plot road elevation grid cross or longitudinal sections. Finally, this tool-suite can visualize the CRG file analysis, showing the comparison of two CRG roads or showing the peaks of a road surface [VIR_10].

3.3.2. Coordinate systems, transformations

To easily work with roads, described as grid surfaces, OpenCRG® manages both x-y-z and u-v-z coordinate system. Hence, it benefits from an evaluation tool to change between coordinate systems.

The strategy to transform the Curved Regular Grid road into whatever coordinate system is based on the transformation of the reference line data. Therefore, when the open source generates a road, its reference line is described – in the CRG structure – in both coordinate systems. Necessary and sufficient parameter fields define the u-v coordinates of the reference line and a look-up table has the node positions in the x-y coordinate system [VIR_11].

3.3.2.1. From Cartesian to Curved coordinates

To transform a P point from *Cartesian coordinates* to *surface coordinates* requires first the identification of the road nodes that surround P (see Figure 3.9) and then the calculation of its u and v coordinates. The flow diagram Figure 3.10 explains schematically how a P point is transformed for a given CRG data structure [DAI_10] [VIR_11].

The starting point of the algorithm is a P point to transform its coordinate system and a CRG road. Actually from the CRG data structure only some head fields (the u and x, y origin and end of the reference line, the gap in u direction), the reference line description in *Cartesian coordinates*, and some trigonometric values (the sine and cosine of start and end value of *heading angle*).

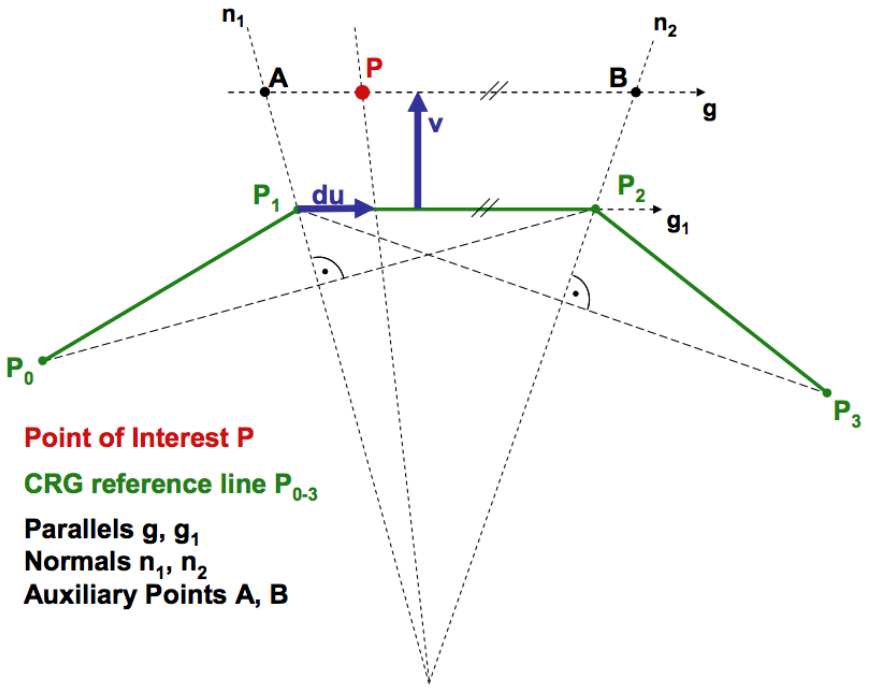


Figure 3.9. Geometric relations of the surroundings of a point P being transformed from Cartesian to surface coordinates [DAI_10].

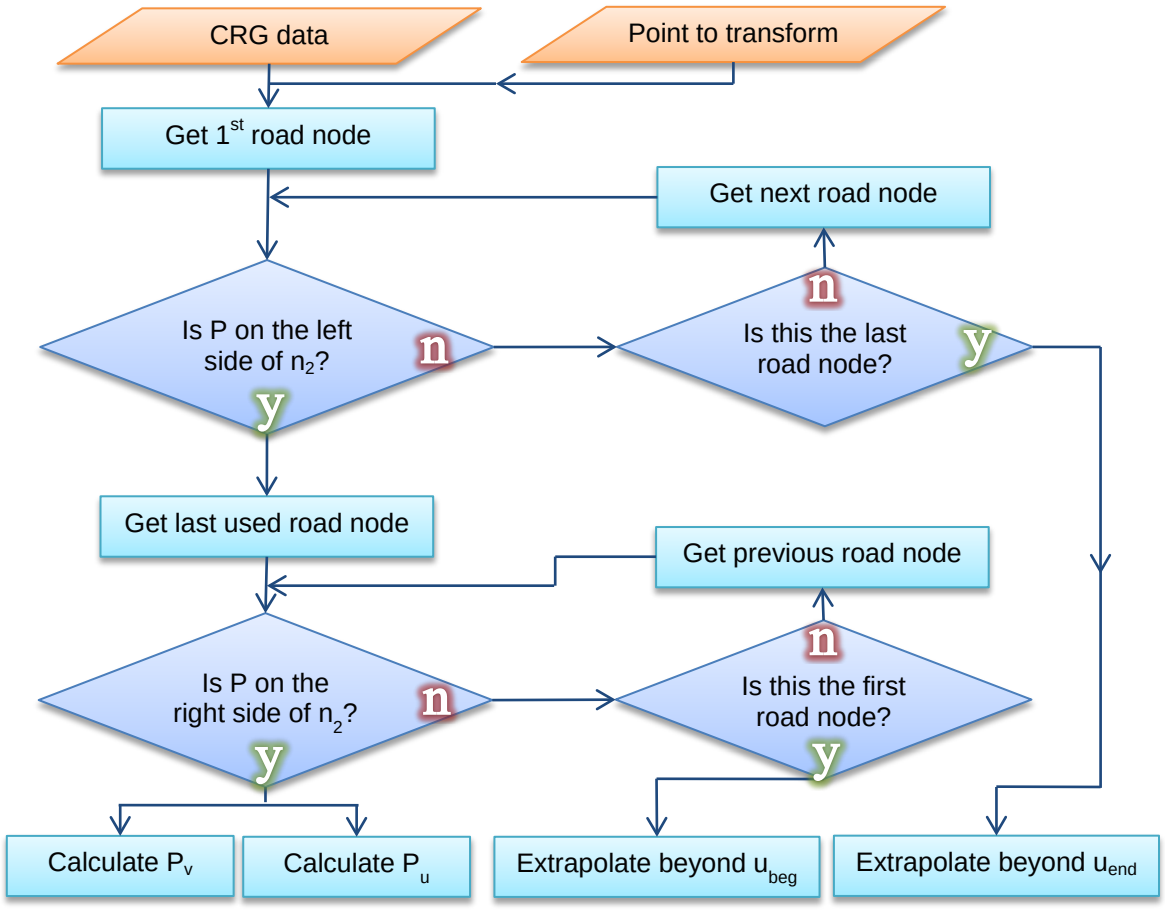


Figure 3.10. OpenCRG© transform evaluation tool x-y to u-v flow diagram [VIR_11].

Afterwards, it starts the search of the P_0 P_1 P_2 P_3 points that surround P in two steps. First, a forward search detects when P point is aligned on the left side of n_2 (see Figure 3.9) analysing the road nodes from the beginning to the road end until reach the condition. Second, a backward search finds out the points from CRL where n_1 (see Figure 3.9 again) is displayed on the right hand side of P, going backwards from the last analysed group of nodes in the forward search. The first condition (to be on the left side of n_2) may be calculated as the unnormalized projection of $\overline{P-P_2}$ on $\overline{P_3-P_1}$:

$$\langle \overline{P-P_2}, \overline{P_3-P_1} \rangle \leq 0 \qquad \textit{(Eq. 3.12)}$$

If P is aligned on the right side of n_1 requires the unnormalized projection of $\overline{P-P_1}$ on $\overline{P_2-P_0}$:

$$\langle \overline{P-P_1}, \overline{P_2-P_0} \rangle \geq 0 \qquad \textit{(Eq. 3.13)}$$

The algorithm second phase evaluates the result. If some condition is false for the whole road means P point is located beyond the start or end point of the road, respectively for first and second condition. If so, simple transformations are applied. Eq. 3.14 and Eq. 3.15 extrapolate P beyond the beginning and the end of the reference line respectively.

$$\begin{aligned} P_u &= u_{beg} + cos(\phi_{beg}) \cdot (P_x - x_{beg}) + sin(\phi_{beg}) \cdot (P_y - y_{beg}) \\ P_v &= -sin(\phi_{beg}) \cdot (P_x - x_{beg}) + cos(\phi_{beg}) \cdot (P_y - y_{beg}) \end{aligned} \qquad \textit{(Eq. 3.14)}$$

where, P_x P_y P_u P_v are respectively the P *Cartesian* and *surface coordinates*, x_{beg} and y_{beg} are the *Cartesian coordinates* of the first CRL node, and ϕ_{beg} is the start value of *heading angle*.

$$\begin{aligned} P_u &= u_{end} + cos(\phi_{end}) \cdot (P_x - x_{end}) + sin(\phi_{end}) \cdot (P_y - y_{end}) \\ P_v &= -sin(\phi_{end}) \cdot (P_x - x_{end}) + cos(\phi_{end}) \cdot (P_y - y_{end}) \end{aligned} \qquad \textit{(Eq. 3.15)}$$

where, P_x P_y P_u P_v are respectively the P *Cartesian* and *surface coordinates*, x_{beg} and y_{beg} are the *Cartesian coordinates* of the last CRL node, and ϕ_{end} is the end value of *heading angle*.

Otherwise, if P point has found within the road surface, v coordinate is evaluated as distance between P point and line $\overline{P_1P_2}$ by calculating the normalized cross product:

$$P_v = \frac{\overline{P_2-P_1} \wedge \overline{P-P_1}}{|P_2-P_1|} \qquad \textit{(Eq. 3.16)}$$

And u coordinate is calculated as the P_1 u value plus δu, with δu as orthogonal projection of $\overline{P-P_1}$ in the direction of $\overline{P_1P_2}$. To simplify calculus, the projection is solved with a cross-multiplication: the relation between distances from P to A, and from A to B is equal to the relation between the length of the projection and the distance from P_1 to P_2.

$$P_u = P_{1_u} + \delta u = P_{1_u} + \frac{|P-A|}{|B-A|}|P_2 - P_1| = \cdots = P_{1_u} + \frac{a}{a-b}\Delta u$$
$$a = \frac{\langle \overline{P_2 - P_0}, \overline{P - P_1} \rangle}{\langle \overline{P_2 - P_0}, \overline{P_2 - P_1} \rangle}\ ; \quad b = \frac{\langle \overline{P_3 - P_1}, \overline{P_2 - P} \rangle}{\langle \overline{P_3 - P_1}, \overline{P_2 - P_1} \rangle} \qquad \textit{(Eq. 3.17)}$$

where, P_{1_u} is the u coordinate of the P_1 point, and Δu is the node increment of the reference line.

It is noteworthy that the previous procedure to transform a P point from *Cartesian coordinates* to *surface coordinates* may be used for any road node as well [VIR_11].

3.3.2.2. From Curved to Cartesian coordinates

To transform the components of a P point, from *surface coordinates* to *Cartesian coordinates*, requires three steps: first to locate nodes that surround P (see Figure 3.11) and calculate some normal vectors, second to compute the A and B auxiliary points, and third to calculate the P *Cartesian coordinates*. The flow diagram in Figure 3.12 shows the procedure to transform a P point for a given CRG data structure [DAI_10] [VIR_11].

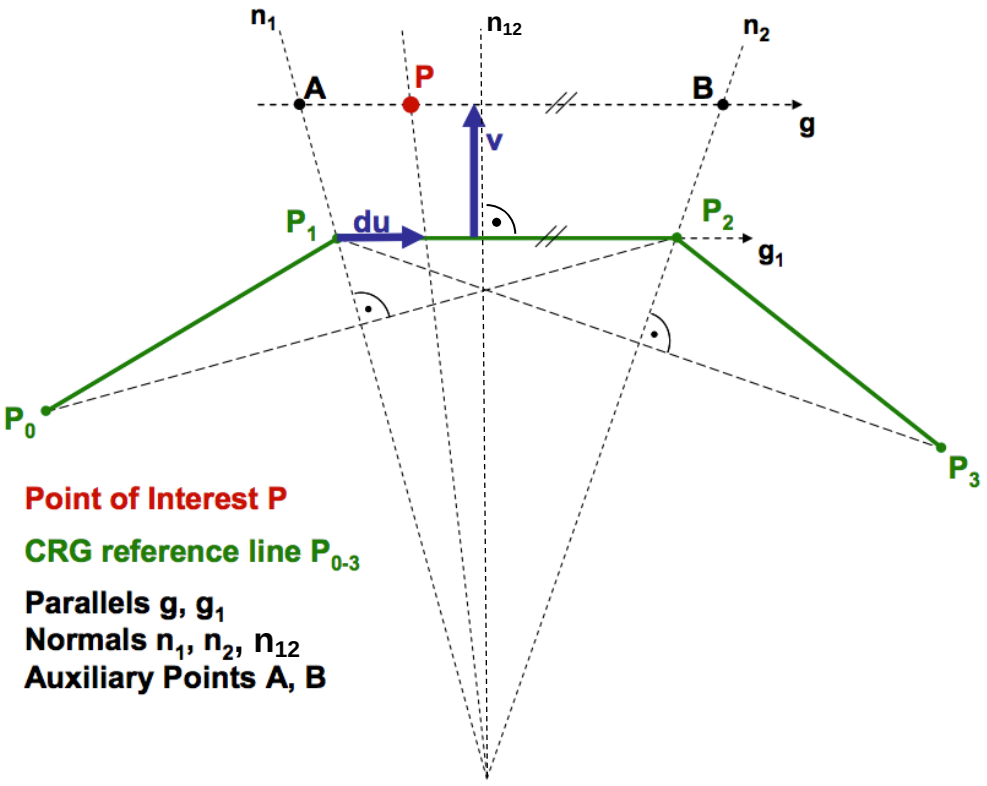


Figure 3.11. Geometric relations of the surroundings of a point P being transformed from surface coordinates to Cartesian coordinates [DAI_10].

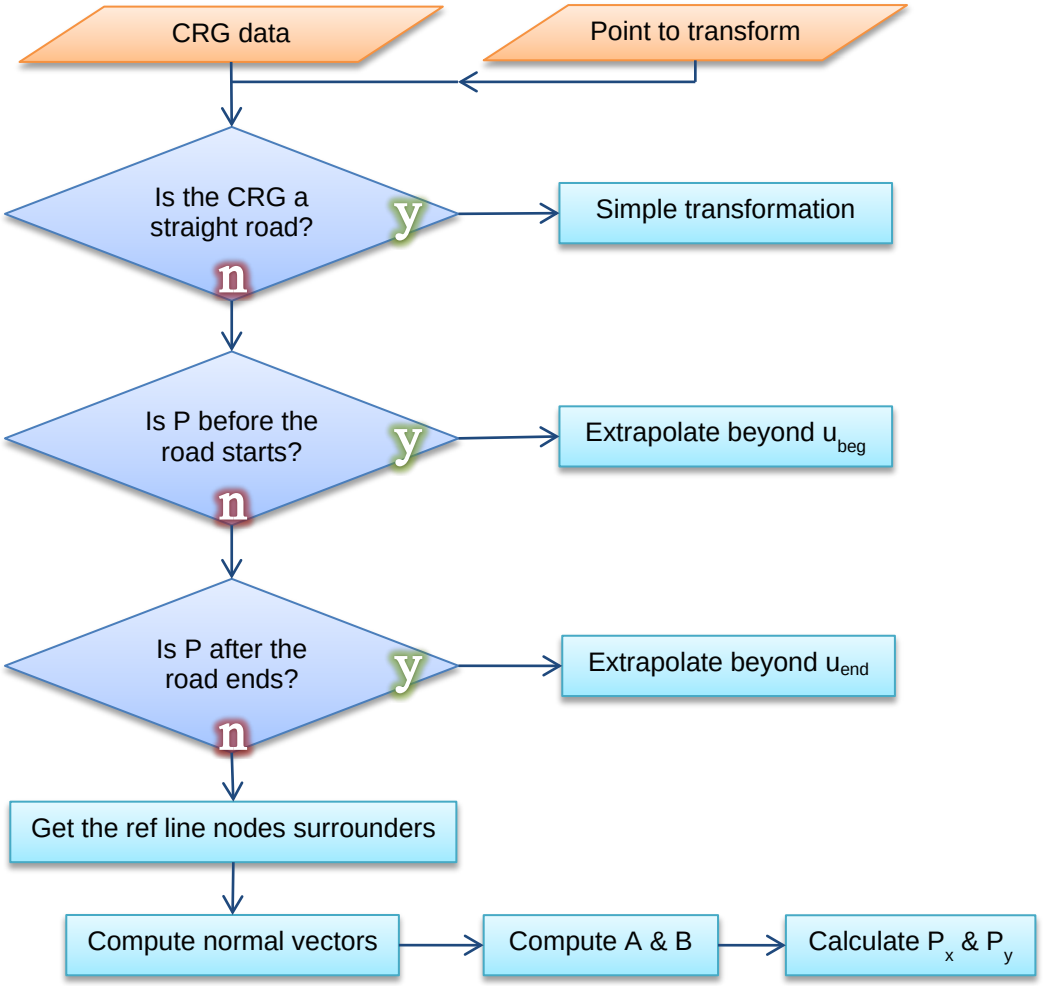


Figure 3.12. OpenCRG© transform evaluation tool u-v to x-y flow diagram [VIR_11].

As in the transformation from *Cartesians* to *surface coordinates* the algorithm starts with the same data: a P point to transform its coordinate system and a CRG road, but from the CRG data structure just some information is needed (the u and x, y origin and end of the reference line, the gap in u direction, the reference line description in *Cartesian coordinates*, and the sine and cosine of start and end value of *heading angle*).

A preliminary step determine the existence of a straight reference line, and if P, the point to transform, is located before or after the road. If it is the case the algorithm finishes applying simple transformations. In case of a straight road the procedure uses these following equations:

$$P_x = x_{beg} + cos(\phi_{beg}) \cdot (P_u - u_{beg}) - sin(\phi_{beg}) \cdot P_v$$
$$P_y = y_{beg} + sin(\phi_{beg}) \cdot (P_u - u_{beg}) + cos(\phi_{beg}) \cdot P_v \qquad \textit{(Eq. 3.18)}$$

where, P_x P_y P_u P_v are respectively the P *Cartesian* and *surface coordinates*, x_{beg}, y_{beg},u_{beg}, and v_{beg} are the *Cartesian* and *surface coordinates* of the first CRL node, and ϕ_{beg} is the start value of *heading angle*.

If P_u value is less than u_{beg} then the point is located before the road, so the following extrapolation beyond the road is executed:

$$P_x = x_{beg} + cos(\phi_{beg}) \cdot (P_u - u_{beg}) - sin(\phi_{beg}) \cdot P_v$$
$$P_y = y_{beg} + sin(\phi_{beg}) \cdot (P_u - u_{beg}) + cos(\phi_{beg}) \cdot P_v \quad \textit{(Eq. 3.19)}$$

In case P is located after the road, computes its *Cartesian coordinates*.

$$P_x = x_{end} + cos(\phi_{beg}) \cdot (P_u - u_{end}) - sin(\phi_{beg}) \cdot P_v$$
$$P_y = y_{end} + sin(\phi_{beg}) \cdot (P_u - u_{end}) + cos(\phi_{beg}) \cdot P_v \quad \textit{(Eq. 3.20)}$$

with x_{end}, y_{end},u_{end}, and v_{end} as the *Cartesian* and *surface coordinates* of the last CRL node.

Because of the regular grid characteristics, it is simple to get the nodes from the reference line that surround P. Thus the first phase of the algorithm is to compute A and B (see Figure 3.11) after calculate the normal vectors $\overline{n_1}$, $\overline{n_2}$, and $\overline{n_{12}}$. Normal 1 is the perpendicular line to $\overline{P_0P_2}$, $\overline{n_2}$ is the normal to $\overline{P_1P_3}$, and $\overline{n_{12}}$ is the perpendicular line to $\overline{P_1P_2}$. A and B auxiliary points are the interjection of the P_v meters parallel to $\overline{P_1P_2}$ with $\overline{n_1}$ through P_1, and with $\overline{n_2}$ through P_2 respectively.

$$A = P_1 + P_v \frac{\overline{n_1}}{\overline{n_1} \cdot \overline{n_{12}}}$$
$$B = P_2 + P_v \frac{\overline{n_2}}{\overline{n_2} \cdot \overline{n_{12}}} \quad \textit{(Eq. 3.21)}$$

Finally, the P *Cartesian coordinates* are calculated with the composition of A plus an increment on the $\overline{AB}$ line:

$$P = A + \left(P_u - P_{1_u}\right) \cdot (\overline{B - A}) \quad \textit{(Eq. 3.22)}$$

where P contains both x and y coordinates, and P_{1_u} is the u coordinate of the node P_1.

As in the previous procedure, it can be extrapolated to transform road node [VIR_11].

3.4. Synergies OpenCRG® – MOVES²

In this work, OpenCRG® is integrated in MOVES² to generate roads. Consequently, synergies from both approaches have used to add new functionalities. On the one hand, main features of OpenCRG® are implemented in the MOVES². These features are the road generation as CRG data structure and some OpenCRG® functionalities, such as visualization tools and evaluation tools to get the road node elevation with and without *banking* and *slope* definitions and to transform the road from the *surface* to the *Cartesian coordinate system*. Despite the fact that just main features of OpenCRG® are implemented in MOVES², most of the uncovered functionalities may be applied manually after the road generation or can be integrated in further work. For example it is possible to cut or limit a given

road, or to margin or compare it with a second road, or implement OpenCRG® modifiers and options to operate on CRG data created with MOVES².

On the other hand, new specific panels implemented in the MOVES² GUI have improved the user-friendliness to create roads (see Appendix A). Although this part of the GUI performs just one feature (generatioin of roads), the idea of flexibility is kept, since it can create roads with any horizontal and vertical alignment. A tutorial of how to create and visualize road alignments can be found in Appendix B.

Through the integration of OpenCRG® in MOVES², new functionalities were implemented to define aspects that are linked with roads (e.g. obstacle perturbations, road roughness, road friction coefficient) allowing the creation of almost any test track. The result is the possibility to simulate and analyse the vehicle body motion when simulating a driving on any test track.

The term *blank CRG road* distinguishes the roads generated just using OpenCRG® from the roads generated using OpenCRG® and some additional functionality of MOVES² presented in this sub-chapter.

3.4.1. Road perturbations: obstacles

Because of the interest to analyse the vertical and horizontal body motion when a vehicle travels on predefined perturbations, the obstacle-builder of MOVES², designed in this work, can add in a simple way and as many times as necessary any kind of obstacle to the road.

The procedure of the obstacle-builder to incorporate a specific obstacle in the CRG road entails the selection of a user-defined 2-D perturbation profile from the corresponding database-library and its extrusion in u or v direction. The obstacle elevation is then superposed to the altitude of the road surface. Through the use of the Simulink block *Signal Builder* – of MATLAB/Simulink® in the GUI of the MOVES² – any perturbation profile can be generated. Furthermore, when a profile is selected, it is possible to use just a section of it and apply a scale factor that amplifies or reduces the altitude. Consequently, a single profile can establish different perturbations. As seen before, *surface coordinates* define the extrusion in u or v direction. The advantages of this u-v road representation are used here since a description in x-y coordinates would complicate the definition of the desired perturbations on a curved road – a sort of polygons (e.g. triangles, trapezoids, diagonal stripes) would be needed. Under those circumstances, the origin and the cross section length of the extrusion, as well as the extrusion width in u or v direction set the region covered by the obstacle. In case of length differences between perturbation profile and cross section, this second parameter causes a profile readjustment stretching or narrowing the profile to be used to create the perturbation. Finally, a visualization tool may plot the resulting road.

Figure 3.13 shows the addition of an obstacle over a 7 meters straight *blank CRG road*, where the obstacle profile is extruded from the point (0, -1) in v direction with a cross section length of 7 meters

and a extrusion width of 2 meters. This setup covers the whole road surface, however the obstacle profile contains null elevation values so, in this case, it results on a double bumped road.

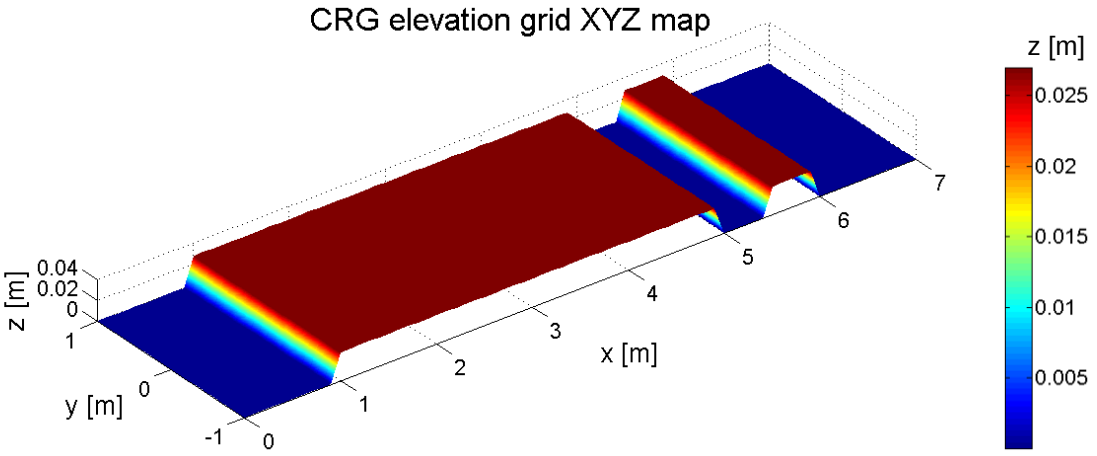


Figure 3.13. Road with bumps composed of a flat and straight road and a double bumped perturbation profile extruded in v direction. Created with the obstacle-builder developed in this work.

3.4.2. Road roughness

The interest of some vehicle dynamic simulations may lie in the analysis of vehicle motions over roads with defined or varying roughness according to the International Standard ISO 8608 [ISO_95]. Therefore it is possible to add a stochastic roughness category (from A to H) to the entire road [ROJ_11b].

The following figure illustrates a road with roughness category C, which characterizes a very rough road – rarely seen in Europe – often used to evaluate ride comfort in vehicle simulations.

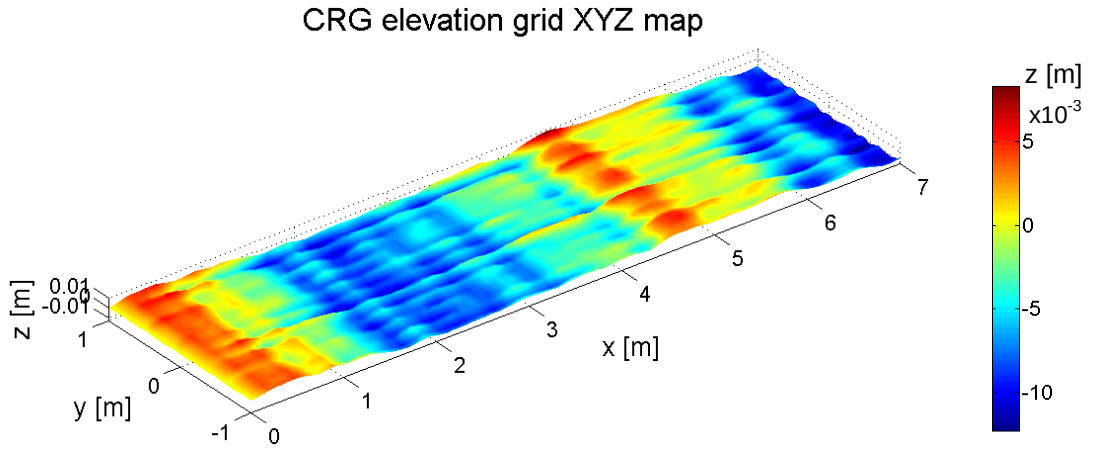


Figure 3.14. Road with roughness category C.

3.4.3. Friction coefficient

Depending on the modelling complexity of the tire model, the friction coefficient of the road may influence the vehicle model behaviour during the simulation. This context justifies the present functionality, which defines different friction coefficients on the road. Consequently, it is possible to simulate manoeuvres in which the vehicle drives over surfaces with non-constant μ. An example is the split-μ manoeuvre, where a vehicle on a road with the coefficient of friction split side-to-side (one side has a high and the other side has a low value of the friction coefficient, μ) [NIE_11].

As the obstacles, for reasons of easy road location *surface coordinates* are used to determine road sections, in which a specific behaviour of the friction coefficient μ is defined. However, the default friction coefficient is equal to one – the maximum friction coefficient, μ_{max}, of a tire on a *dry asphalt* [GUS_98].

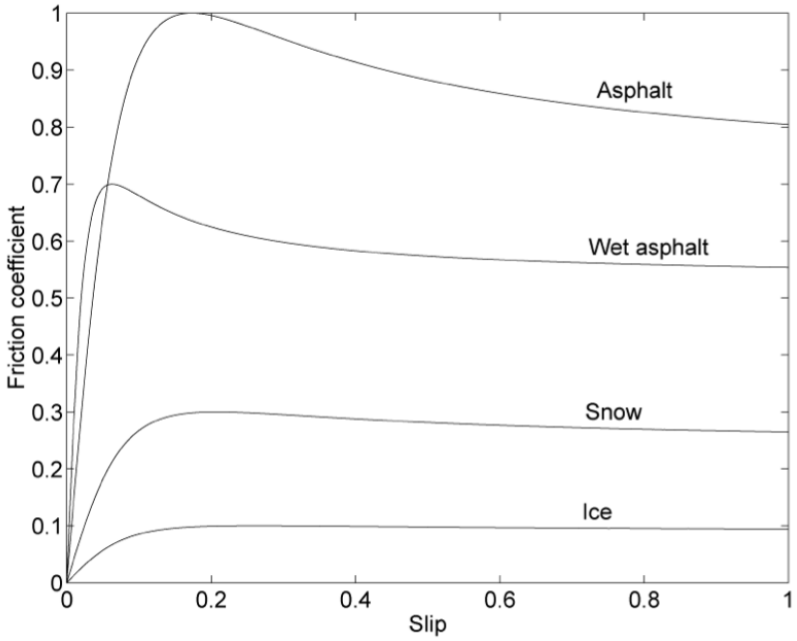


Figure 3.15. Schematic μ-slip curves for different road conditions, with exaggerated initial slope differences [GUS_98].

3.5. The usage of 3-D roads in 2-D vehicle models

The simulation environment required for two-dimensional vehicle models mainly corresponds to a two-dimensional road. In fact this road is a road signal that gives altitude at the *driven distance* positions.

As stated in sub-chapter 2.3, MOVES^2 also includes two-dimensional vehicle models such as the quarter or the single track vehicle model. Interests in simulate a certain manoeuvre using a progressive modelling complexity at the vehicle level, made necessary the usability of three-dimensional roads within a two-dimensional simulation environments.

Therefore, the road signal is generated based on the wheel contacts for the desired trajectory (see sub-chapter 2.5) in the three-dimensional road; the desired trajectory is shifted right and left according to the median wheelbase specified in the vehicle definition.

Figure 3.16 illustrates a three-dimensional straight road with a sinus perturbation. In this case the desired perturbation, shown in green colour, follows the road midline. The wheel contacts, highlighted in red colour, are calculated by shifting the trajectory right and left. Both wheel contacts are shown in Figure 3.17 as road signals, relating *driven distance* and altitude together.

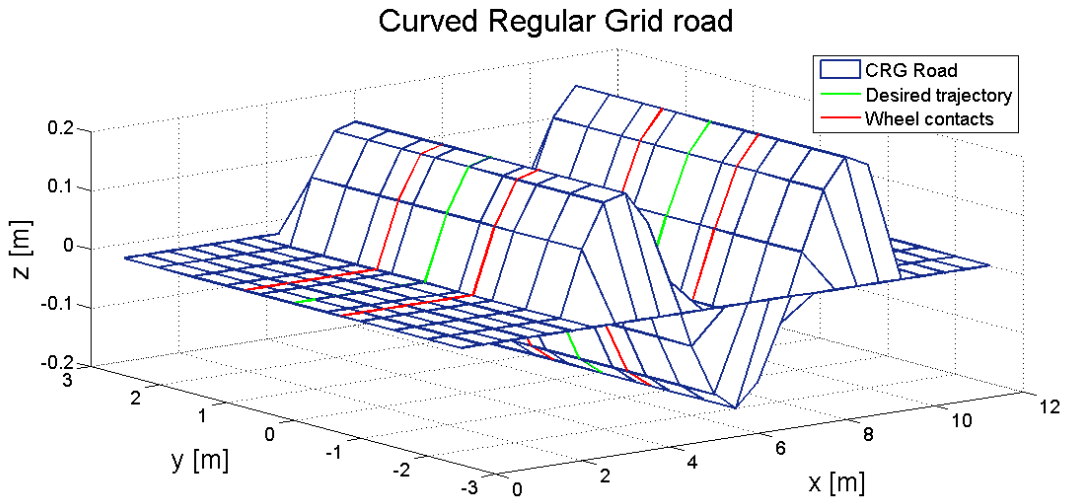


Figure 3.16. Wheel contacts for a specified trajectory in a three-dimensional road.

Figure 3.17. Wheel contacts from the road shown in Figure 3.16.

4. ANIMATION MODULE

This chapter exposes the animation module implemented in MOVES[2] in this work and introduces the source and the language code to create an animation. The chapter concludes with a summarization of the main features of the animation module.

The advantages of generate an animation of the vehicle motion may depend on the user knowledge and experience. Non-advanced users may have difficulties to find the relevant variable results, to relate different variables, to generate appropriated plots, and to interpret them quickly. The animation gives to non-advanced users a general idea about how the vehicle is traveling along the simulation environment. For advanced users, the animation generation is not necessary but useful; thus it is a complement of the simulation results and the output plots. For example, thanks to the animation module, it is possible to understand the influence of more than one parameter and to analyse a situation involving multiple cars. The generation of an animation of multiple cars is a manual procedure that merges two animation files from the same road. Appendix D details this manual procedure.

In this work, MOVES[2] has integrated the toolbox Simulink 3D Animation™ [5] to generate animations. This toolbox uses Simulink data to animate the vehicle motion in a three-dimensional virtual reality environment. Thanks to the VRML viewer of the Simulink 3D Animation™, it is possible to display the virtual world from any point of view, as well as to navigate through it [MAT_10].

4.1. Virtual Reality Modelling Language

Virtual Reality Modelling Language [ISO_97] – VRML – is an ISO standard that is open and text-based. It codes the virtual world and also the animation recording.

4.1.1. VRML coordinate system

VRML uses the coordinate system in which the y axis points upward and x and z axis determine the horizontal plane, applying the *right-hand rule*. The x axis corresponds to the driving direction at the starting point (see Figure 4.1).

[5] Since MATLAB® R2009a, Virtual Reality Toolbox™ has been renamed as Simulink 3D Animation™.

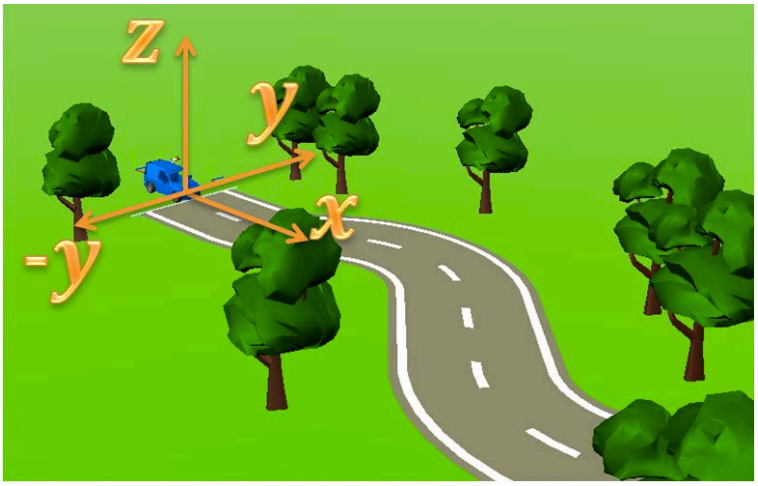


Figure 4.1. VRML coordinate system [MAT_10].

4.1.2. The VRML file

VRML files can contain five different types of component: VRML header, comments, nodes, prototypes, and routes. The VRML header specifies that a file corresponds to a VRML file, so therefore it must appear always on the first line. Comment lines are distinguished by # at the beginning of the line. Nodes describe how the virtual world looks like. Nodes group shapes, transformations, timers, interpolators, sensors, fog, scripts, etc. Shape nodes group geometrical shapes and grid shapes. Although both of them share some characteristics (appearance, material and texture), geometrical shapes are defined by parameters (e.g. radius, height) and grid shapes are generated by a grid of points. By them a road with lane lines and traffic cones can appear at any space point (using appropriate transformations to move and rotate shape nodes) within a virtual world. Moreover, it is possible to establish node hierarchies, so transformations may affect groups of nodes. Prototypes can create new node types. Routes may make the world dynamic, linking two nodes by an *event in* and an *event out*. For example, clicking on a shape the light turns on, or if there is a collision with a shape a sound is played [AME_97] [ISO_97].

VRML editors can create and edit virtual worlds. Simulink 3D Animation™ includes the VRML editor Ligos V·Realm™ Builder software [LIG_05]. Moreover, CAD software can export models to the VRML code [AME_97] [MAT_10]. Appendix C describes how CATIA models can be exported to the VRML file format.

4.2. The VRML scene

Besides vehicle motion data, the animation generation requires a virtual world where the full environment definition appears on it. For modularity reasons, the VRML scene generates the virtual world, connecting the different environment parts together. As the diagram from Figure 4.2 illustrates, these linked environment parts are the vehicle shape and the scenario world, which in turn links the unanimated objects. They are stored in their corresponding VRML database-libraries.

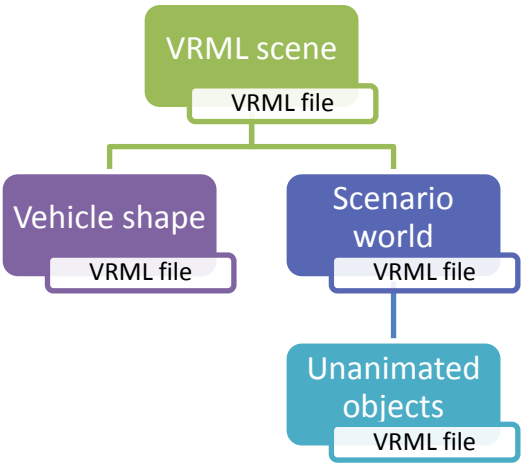


Figure 4.2. The VRML file hierarchy of the VRML scene

Apart from the linkage of these files, the VRML scene codes the visualization of a steering wheel located above the vehicle shape, the four wheels with their contact force visualized by three vectors in x, y and z direction each, and the backlights of the vehicle. Moreover, based on the data from the Simulink vehicle model it makes possible the animation of these objects as well as the vehicle shape during the animation. Figure 4.3 illustrates an animation where a vehicle drives on a left curved road. It shows the steering wheel turning left, the force vectors from the wheel contacts and the front wheel slightly rotated to the left.

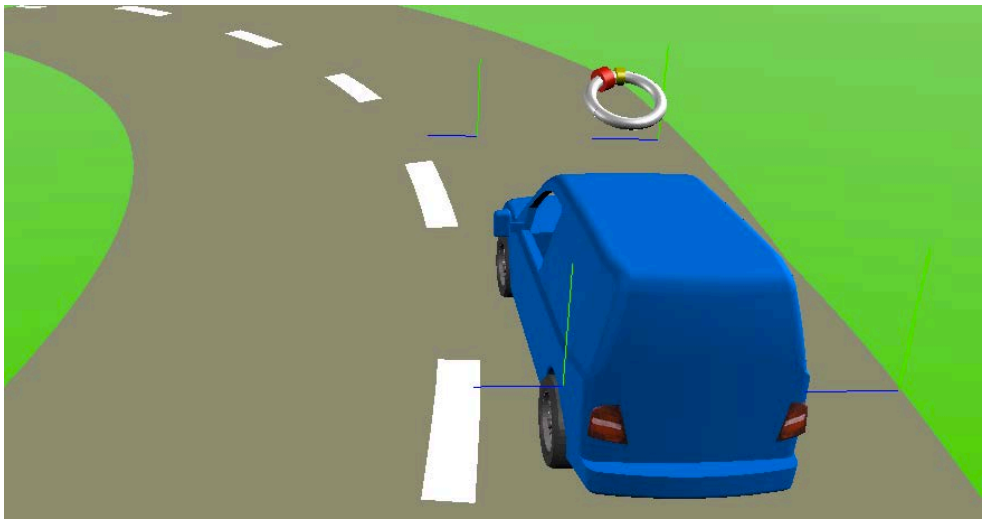


Figure 4.3. Animated objects during the animation coded within the VRML scene. Screenshot of the visualization feature implemented in this work.

Specific parts of the GUI of MOVES[2] can select those files linked to the VRML scene, and edit their database-libraries (see Appendix A).

4.2.1. Vehicle shape

The vehicle shapes animate the vehicle motion within the scenario world. To ensure concordance with vehicle definition set to simulate, the vehicle shape is adjustable. The vehicle shape VRML database-library stores the available VRML vehicle shapes.

Figure 4.4 shows a vehicle shape designed at the Institute of Automotive Engineering at Graz University of Technology, based on a vehicle existing in the market. In this case, this shape was designed in CATIA® V5 using the tool to export parts to the VRML format (see Appendix C).

Figure 4.4. Vehicle shape designed in CATIA® V5 at the Institute of Automotive Engineering at Graz University of Technology.

4.2.2. Scenario world

Based on the interest to highlight specific road sections during the animation, the scenario world definition combines data from road specifications (the road profile and the friction coefficient definition), used to execute simulations, with additional information from the road appearance, grouping obstacles, friction coefficient sections, and lane lines. Moreover, for a better realistic scenario, the scenario world places the unanimated objects set on its parameter script. The scenario world files are stored in the scenario world VRML database-library.

Each obstacle and each friction coefficient section can be individually coloured and for better visualization a set of colour maps, with the colour gradient based on the elevation coordinate, can paint road perturbations. Lane lines may mark lane boundaries along the road, therefore the v coordinate places them. Moreover, every line has its own geometrical characteristics. These characteristics establish line width and dashed property. Lines composed of dashes have custom gap and line length.

Making use of the linkage of the VRML code, the scenario world adds the specified unanimated objects on it. Thus, objects are placed in the scenario world in the specified place with the specified orientation.

The following figure shows the scenario world of a split-μ manoeuvre [FOR_98], In this case, the road appearance establishes three road lanes and a blue area, suggesting an ice surface, with the low friction coefficient value. Regarding the road lanes, dashed lines indicate a lane division and solid lines limit the road.

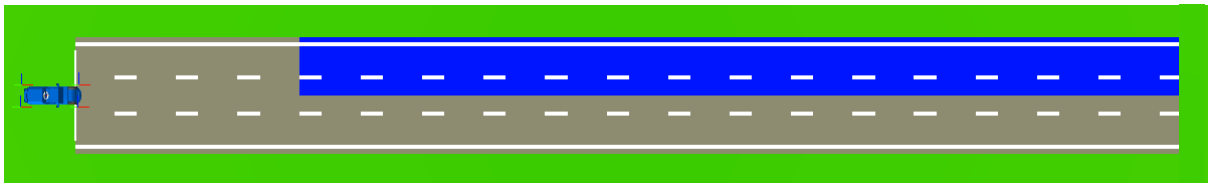


Figure 4.5. Scenario world of a split-mu manoeuvre with lane lines and the road section side of the low μ coloured. Screenshot of the visualization feature implemented in this work.

4.2.3. Unanimated objects

Required objects for specific manoeuvres or just decorative objects can be placed in the scene. Both *Cartesian* and *surface coordinates* can locate objects. The default elevation of an object corresponds to the road altitude at the object's position, although it is possible to place it above or below this point. Additionally, rotations around the x, y, and z axis can turn each object according to its pre-set orientation.

The GUI of MOVES[2] allows to view and edit the objects from the unanimated objects database-library, making use of the Ligos V·Realm™ Builder software [LIG_05], the VRML editor included in Simulink 3D Animation™. Figure 4.6 previews the unanimated object *tree* through four viewpoints, showing its pre-set orientation, and the top, right, and front views.

Figure 4.6. The preview of Ligos V·Realm Builder for the unanimated object tree.

4.3. Animation performance

For a better experience concerning animation performance, setups and functionalities offer some special options.

One of the available options is to display simulation data through the VRML viewer when the animation runs. Actually the data is displayed as text on a corner of the viewer window, and it can be static, dynamic or a mixture of both. At any case, the dynamic data corresponds to Simulink variables coming from the simulation. For example, the vehicle speed can be displayed in the animation viewer during the simulation.

Another functionality added in the animation module is to show the vehicle real trajectory. Then, through the VRML viewer, target and real trajectory lines are easily comparable with just a quick overview on the scene.

A manual procedure can create animations with multiple cars. Thus, the animation requires simulated vehicle motion from more than one simulation configured with the same scenario. This animation type allows comparing multiple simulation parameters at once. Appendix D details this manual procedure.

Different setups can be used to enable the animation generation at specific phases of the simulation process – carried out in MOVES^2 –, either during the simulation execution or afterwards, in a post-processing function. In the first case, the animation is automatically stored in the temporary folder. In the second case, it is possible to store the animation either in the temporary folder or in the results folder. However, the VRML viewer allows to save the animation in the desired folder. The available formats to save an animation are the three-dimensional VRML and the two-dimensional AVI format files.

5. SCENARIO-BUILDING MODULE

The previous two chapters presented an accurate description of the road- and the obstacle-builder as well as of the animation module that have been implemented in this work in MOVES^2. Also, the previous sub-chapter 2.5 exposed the manoeuvre-building module also implemented in this work in MOVES^2. Although each chapter refers to different environment characteristics, sub-chapter 2.5 details how to create the desired trajectory, Chapter 3 describes the road generation and Chapter 4 explains the animation performance. These features compose the so-called scenario.

This chapter describes the applied methodology to manage the variables and the information concerning a simulation scenario and explains how the parental relations within the scenario are established.

5.1. Scenario modularity

Following the standardized programming methodology applied in MOVES^2, a library from *Parameters library* collects the scenario parameter script files (see Figure 2.2). Because of the complexity of the simulation scenario, each parameter from the scenario parameter script defines one concept of the scenario. Examples of scenario concepts are the road alignment, the unanimated objects placed on the road or the friction coefficient definition of the road. Different sub-libraries from *Parameters library* gather the different concepts of the scenario. Hence, each concept of the scenario is defined by linking a parameter from the scenario parameter script with a parameter script from one of these sub-libraries.

As Figure 5.1 illustrates, the scenario definition uses data from eight different sub-libraries split into three groups: The road specifications collect four sub-libraries to define the road geometry (road alignment, road perturbations, and road roughness) and the friction coefficient of the road. The manoeuvre specifications select the used parameter script from the trajectory sub-library, where the trajectory is set up as described in sub-chapter 2.5. Finally, the animation specifications use three sub-libraries to set the scenario appearance (lane lines, unanimated objects and vehicle shape). If the scenario does not require the information from any of these eight sub-libraries, scenario parameters would select as default an empty parameter script file from the sub-library.

As a result, the scenario parameter script deals with the following modules developed in this work: the scenario-building module, the manoeuvre-building module and the animation module. Specific panels of the GUI of MOVES^2 can change and create scenario parameter script files used by the simulation by generating, modifying or deleting the content of the eight sub-libraries (see Appendix A).

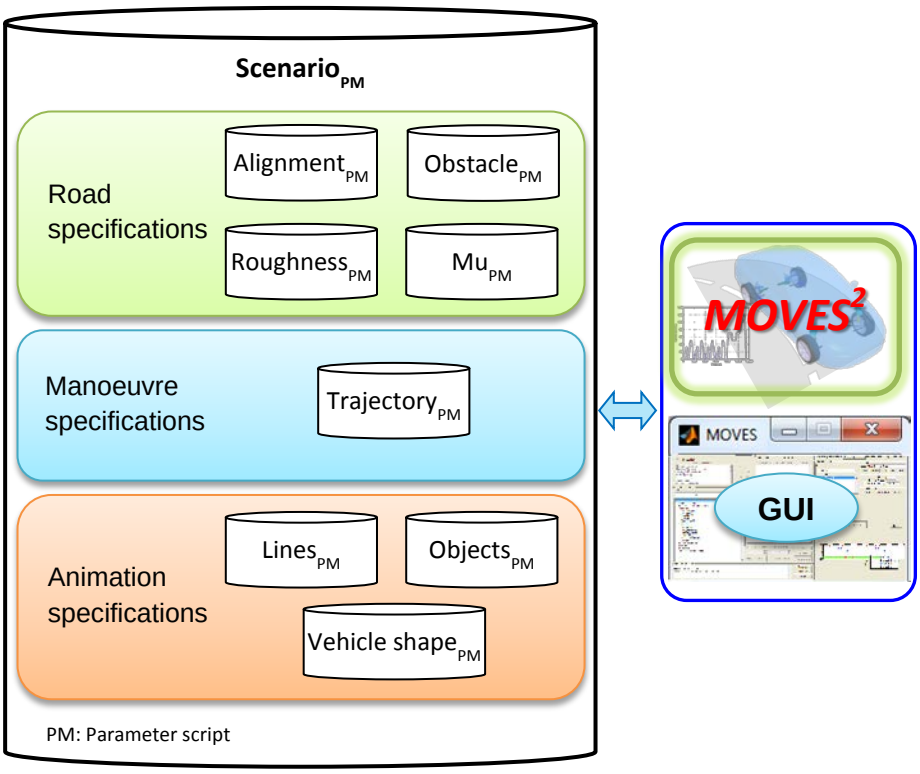


Figure 5.1. The Scenario library and its linked sub-libraries. Specific parts of the GUI of MOVES² can adjust the road, manoeuvre and animation specifications.

5.2. Scenario management

The way to manage the scenario data differs a little bit from the standardized programming methodology. Usually, a pre-processing function should calculate new parameters based on the data of the parameter script that the function is linked with, but the scenario pre-processing function differs from this standard process. Because of this singularity, the scenario pre-processing function is described in the present sub-chapter by the diagram shown in Figure 5.2.

Three main steps configure the scenario pre-processing function: first, it loads the different parameter script files set on the scenario parameter script (as shown in Figure 5.1); second, it assigns the data contained there to the MOVES² input structure; and third, it executes the pre-processing functions associated with the loaded parameter scripts. These pre-processing functions are the road-manoeuvre pre-processing function and the animation pre-processing function. They generate the three-dimensional road together with the desired trajectory for the driver controller, and the virtual world and the animation in case it is active, respectively. Moreover, as shown in Figure 5.3, the scenario pre-processing function uses database-libraries.

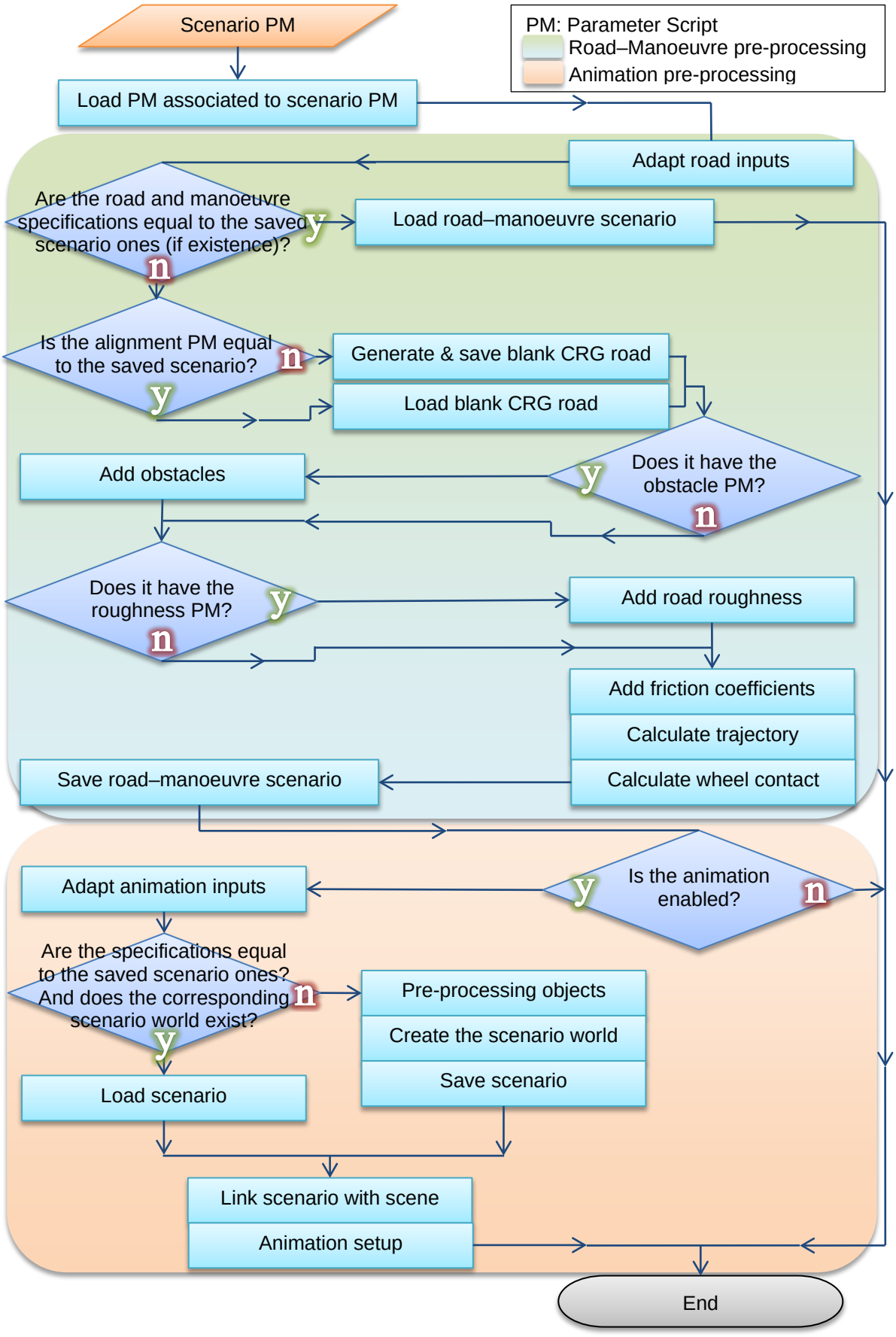


Figure 5.2. Flow diagram of the scenario pre-processing function.

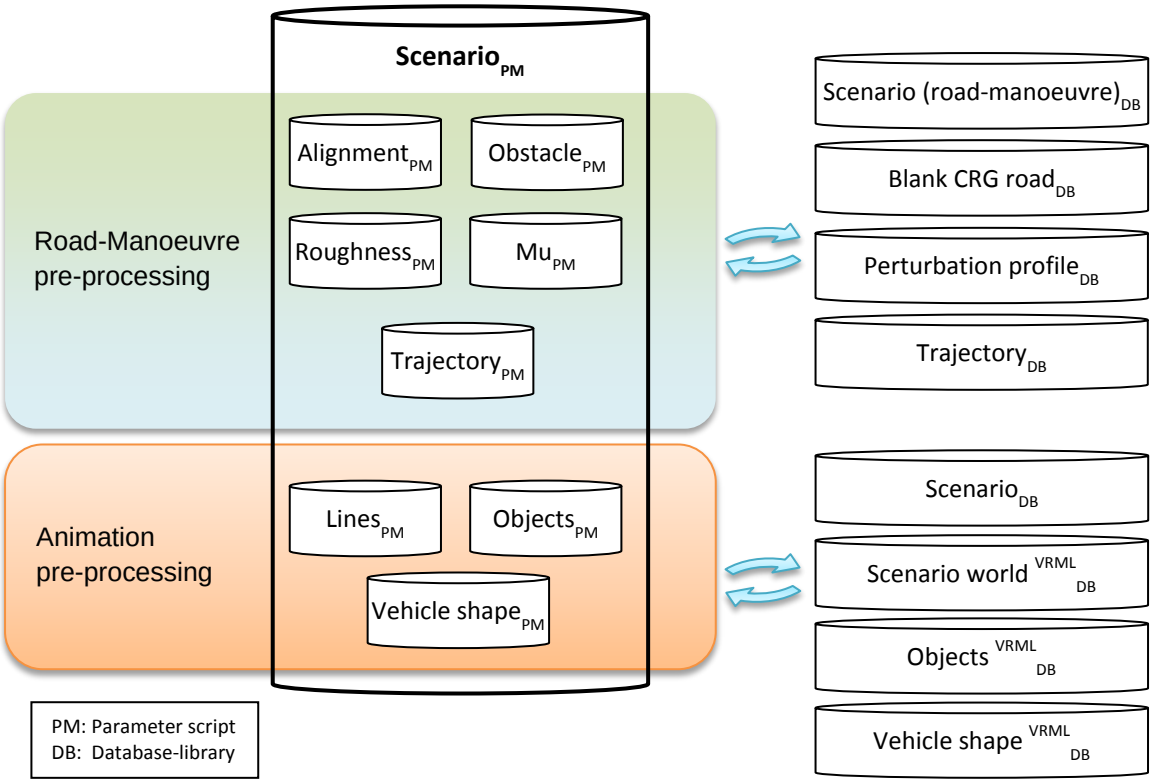


Figure 5.3. The interaction of the scenario pre-processing function with the database-libraries.

To generate the road, the parameters from the road specifications are adapted in a first step: Redundant information from the alignment parameter script is removed from the structure (e.g. a curvature section with null length). Then, to determine exactly the road midline and to avoid conflicts with the function that generates the road roughness, an odd number of nodes in v direction is imposed to the road, by recalculating the Δv – node increment in v direction. The Δu, Δv – node increments in u and v directions respectively – and road width are forced to be multiple of the minimal CRG increment (10^{-3} meters). Later on and because of Nyquist frequency requirements, the Δu is limited to a maximum of 0,313 meters if the roughness is specified. Afterwards, a function creates the polynomials to describe the road *curvature*, *slope* and *banking*. This is done based on the user inputs of the horizontal and vertical road alignment, which defines the so-called *blank CRG road*. As described in sub-chapter 3.4, a blank CRG road is a road without perturbations and roughness. Additional functions called by the scenario pre-processing function adjust the road perturbations and friction coefficient sections to the node grid of the road.

After this adaptation, the algorithm checks if the current road and manoeuvre specifications correspond to some scenario stored in the scenario database-library. In case of match, this scenario is loaded to save time, while it contains the data that this road-manoeuvre pre-processing function calculates, and this road-manoeuvre pre-processing function ends; otherwise the algorithm checks if the current road alignment inputs are the same to some road stored in the blank CRG road database-library. In case of correspondence, the road is loaded to save the road generation time; if not, the new *blank CRG road* is generated and added in its database-library.

According to the loaded parameter scripts, the road is then modified to add road perturbations and road roughness (see sub-chapters 3.4.1 and 3.4.2): The procedure entails the creation of areas with a certain altitude to be superposed to the altitude of the road. The parameters from the obstacles parameter script define this area – with the cross section length and the extrusion width – and its location on the road for each obstacle. In this case, the algorithm loads those set 2-D profile perturbations from their database-library and extrudes them in u or v direction to add them on the road; if the road roughness is specified, the area concerning the road roughness is equal to the road in *surface coordinates*. Then it is created and superposed to the road altitude.

Afterwards, the friction coefficients for each node of the road are assigned to the road structure (see sub-chapter 3.4.3). Then, the target trajectory is defined using the trajectory parameter script. According to the option to create the trajectory that is set in this parameter script, the algorithm uses, as detailed in sub-chapter 2.5, the road database-library, the trajectory database-library or the trajectory functions. If no option is selected, the algorithm selects as the default target trajectory the road midline. Subsequently, the wheel contact lines are calculated to determine the excitation signals for case a two-dimensional vehicle model of MOVES^2 is used, as described in sub-chapter 3.5. Finally all the generated scenario data together with the road and trajectory inputs are saved in the scenario database-library.

If the animation is active the algorithm runs the animation pre-processing function. It first adapts the data from the lines parameter script, which were described in sub-chapter 4.2.2, when the lines of the road are defined out of the road.

Then the algorithm checks two conditions: if some scenario from the scenario database-library has the same parameters as the parameters from the current scenario and if the scenario world associated to the current scenario exists in the scenario world VRML database-library. If both conditions are true the stored scenario is loaded and its parameters are added in the structure; otherwise the objects data is pre-processed, the scenario world file is generated and the entire scenario data is saved in the scenario database-library. This pre-processing process consists in calculate the objects position in *Cartesian coordinates*, since either *surface* or *Cartesian coordinates* can locate them, and in calculate their general rotation axis based on basic x, y, z axis rotations. The scenario world is generated in the VRML code according to the road, manoeuvre and animation specifications, as stated in sub-chapter 4.2.2, and later on it is stored in the scenario world VRML database-library.

Lastly, a function links the VRML scene with the vehicle shape and the scenario world. As commented in sub-chapter 4.2, this linkage is based on the VRML files hierarchy. Another function makes ready the animation according to the inserted setup. As described in sub-chapter 4.3, the available setups enable the animation generation either during or after the simulation execution.

It is noteworthy that the GUI of MOVES^2 does not allow introducing redundant or invalid data concerning the road, manoeuvre and animation specifications, e.g. null curvature section in the road

alignment parameter script and a negative coefficient of friction in the mu parameter script. Moreover, it automatically checks and corrects the introduced areas in the obstacle-building (see sub-chapter 3.4.1) and in the road sections defined in the friction coefficient definition (see sub-chapter 3.4.3). This correction adjusts the areas to the node grid of the current road in the GUI.

6. MODEL MANAGEMENT TOOL

This chapter exposes the developed methodology to handle with the model complexity of systems and components within the model.

As explained in sub-chapter 2.3, thanks to the *block choices*, the simulation environment implemented in MOVES^2 is capable to deal with different model complexities at system and component level. A panel – developed in this work – from the GUI of MOVES² allows the management of the mentioned *block choices*. Consequently, the complexity level required for the simulation can be adjusted with a high level of user-friendliness.

The developed procedure to manage the modelling complexity is based also on the standardized programming methodology; this procedure entails the usage of a model management parameter script from the model management library from *Parameters library* and a pre-processing function (see Figure 2.2). The model management parameter script contains structured data from each *block choice* of the Simulink model: it defines for each *block choice* its corresponding path in the model and its selected choice. Moreover, this parameter script contains additional information to configure the way these *block choices* are shown in the GUI of MOVES^2, as subsequently described. The task of the pre-processing function is to change the selection from those *block choices* whose selection is different from the one defined in the model management parameter script.

The *block choices* are displayed as a tree list on the GUI, keeping certain hierarchy from the Simulink model. To generate the tree list first, the model is analysed to search *block choices* – indicating vehicle systems and components with variable complexity. Afterwards, the found components and systems are arranged according to the model hierarchy.

The diagram from Figure 6.1 illustrates the paths from the *block choices* that configure the modelling complexity concerning the device (in this case the system that supports the engine weight), the springs and dampers, and the tires. In the model, every label corresponds to a *child* subsystem from the previous one. These *block choices* are located within the *Vehicle model* main label. Then, the *Body* label groups the *device model block choice* and the *Wheels* label groups the four wheel subsystems, with the *block choices related to* the spring, damper and tire in each wheel subsystem.

The model repeats the same structure to characterise the subsystems in specific labels. In the figure the repeated structure, shown in red colour, is contained in the wheel subsystem. When a *block choice* appears more than one time within the model, the first subsystem going backwards in their paths that is different is considered the *key label*. The *key label*, shown in green colour in the figure, brings meaning to the repeated structure. For example, the front wheel left has the same structure compared to the front wheel right, but they are different wheels.

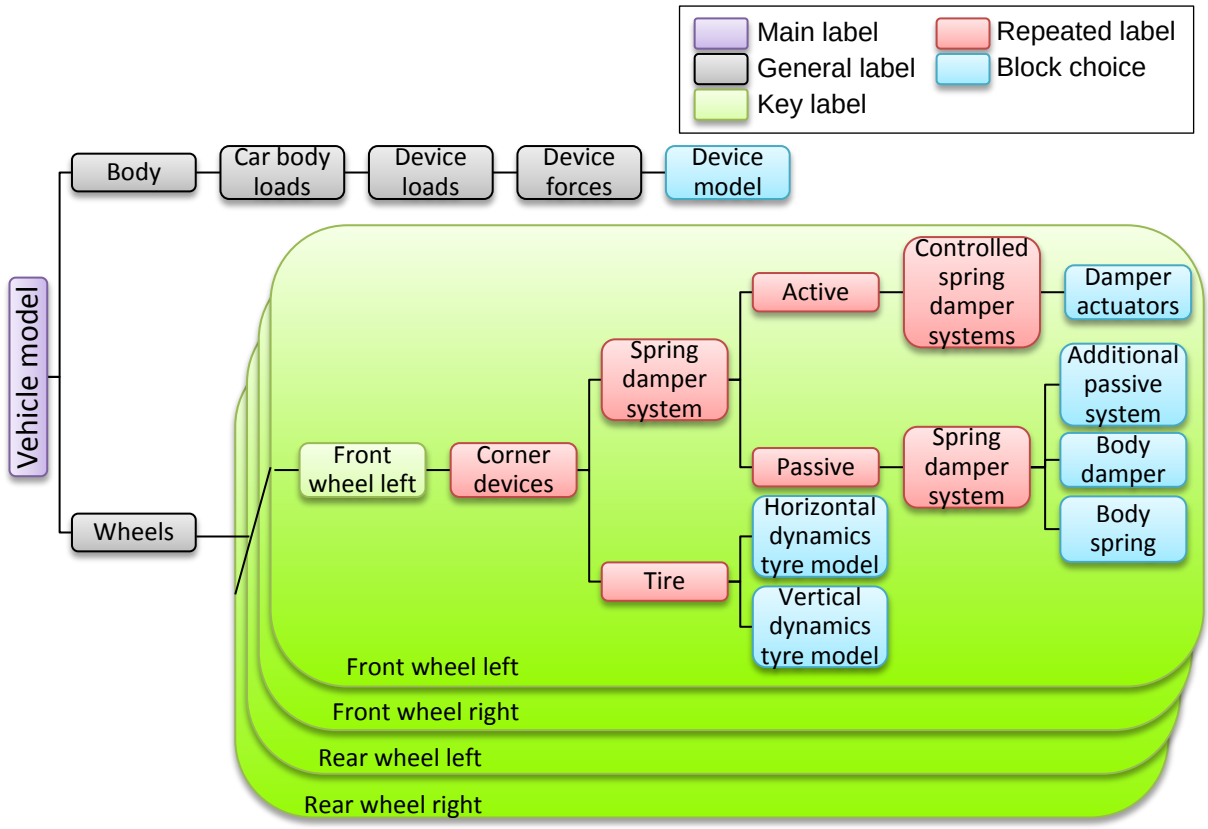


Figure 6.1. Hierarchy of some block choices according to the model implemented in MOVES².

Therefore, to easily identify the *block choices* within a repeated structure as well as to generate a more compact tree list, the repeated labels are removed from the list and the *key labels* are assigned after the *block choices*. In the example from Figure 6.1, the removed labels are shown in red colour.

Figure 6.2 illustrates this rearranged tree list regarding the example shown in Figure 6.1. The *Device model block choice* is unique in the Simulink model thus it keeps the labels from the model. However, the other *block choices* appear four times in the model (one for each wheel) and then their labels are rearranged as explained before. These other *block choices* are used to configure the modelling complexity of the springs, dampers and tires.

Finally, this reorganized list is displayed on the GUI of MOVES² through three tables: two tables to adjust the complexity of the components and systems by editing the *block choices*' selection, and one table showing additional *block choices* that cannot be modified by the user. Therefore, the model management parameter script contains two parameters to split the tree list into three groups: one parameter to select the blocked *block choices* and another parameter to distinguish high and low priority components and systems.

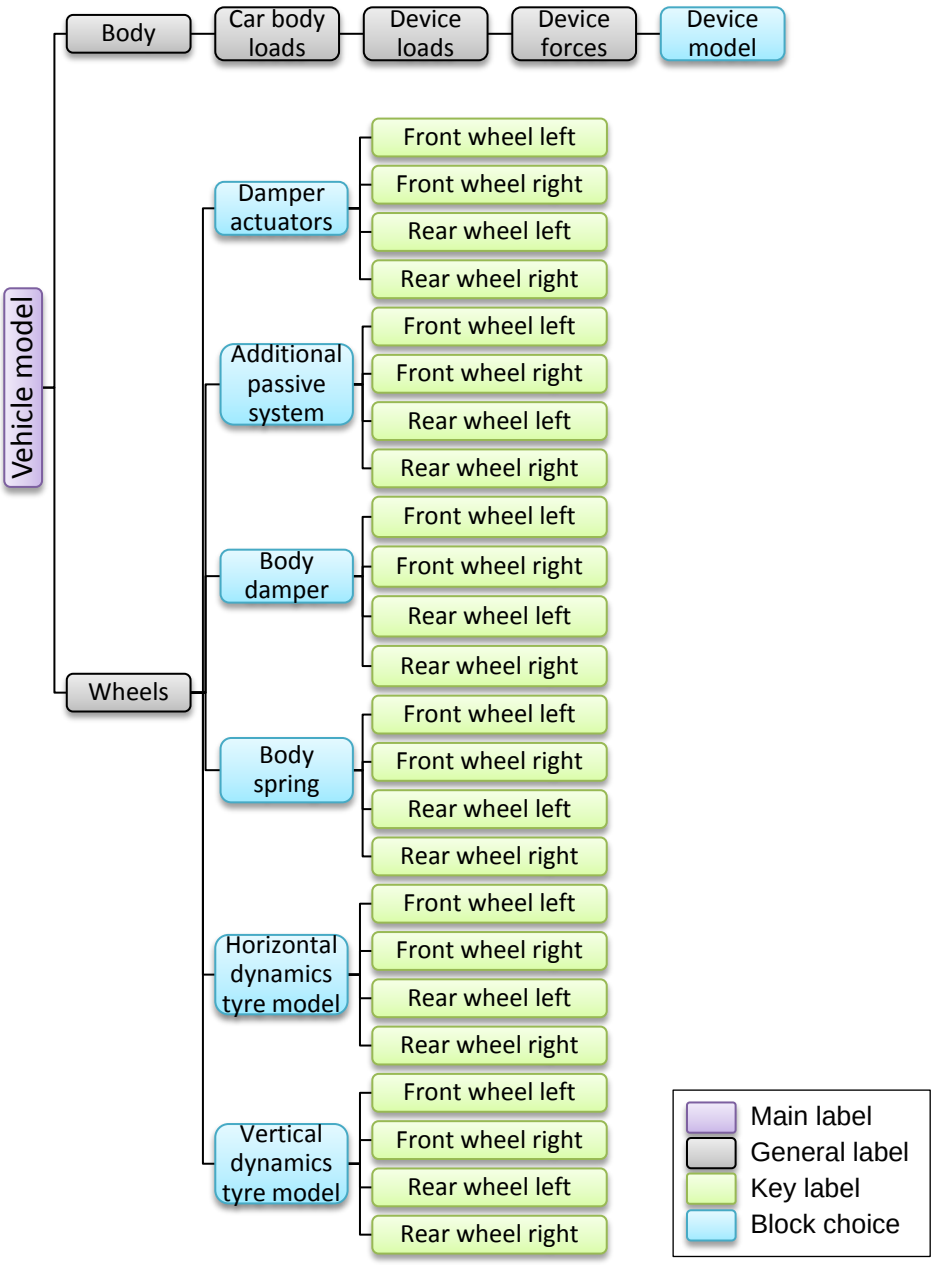


Figure 6.2. Rearranged hierarchy of some block choices according to the model implemented in MOVES².

Figure 6.3 illustrates part of the panel that manages the modelling complexity. The top part of the figure locates the tool to set the priority of the block choices within the model: Choosing a *key word* – a *word* corresponding to a label on the *block choice* path within the model – the *block choices* whose path contains the selected *key word*, are moved to the high priority list. The bottom part of the illustration places the tree block choice list. In Figure 6.3, the low priority list is selected; it displays some reorganized components and systems related with every wheel. In the figure, the four damper actuators are highlighted and disabled at once; the empty block choice has been set.

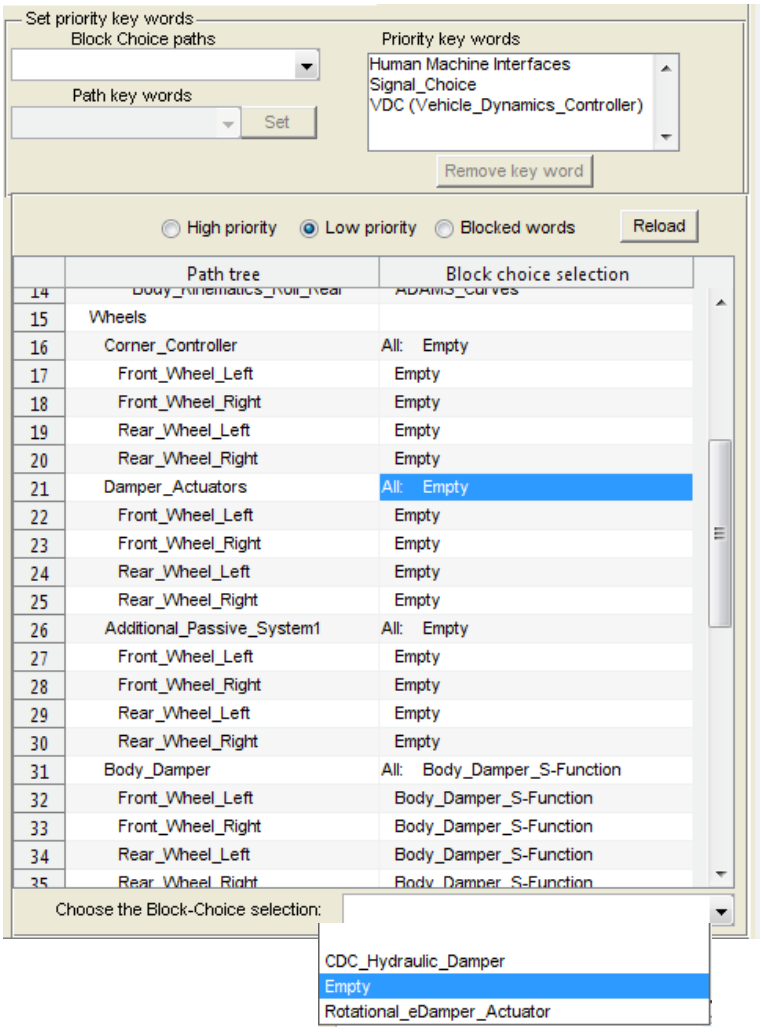


Figure 6.3. A part of the Model management panel showing an example of the reorganized hierarchy of components and systems related with wheels.

The advantage of this modified hierarchy is that the user can to easily identify the selections from every *block choice* at each *key label*, in the example at each wheel. Consequently, the specification of all systems or components grouped under a certain label (e.g. *Corner_Controller* or *Damper_Actuators*) can be changed at once by editing the selection corresponding to the label. Alternatively, the specification of each system or component can be changed by selecting it individually.

7. CONCLUSIONS

In the present work, the vehicle simulation environment MOVES^2, originally developed at the Institute of Automotive Engineering at Graz University of Technology, has been enhanced; the scenario-, and the manoeuvre-building module, together with the animation and the model management tool have been implemented in MOVES^2. This development, completely build up in MATLAB/Simulink®, is in based on the standardized programming methodology of the approach, which has a modular, expandable, and flexible simulation model. Because of these characteristics, MOVES^2 can play a significant role in the early design phase of the vehicle development.

MOVES^2 can be used to model and simulate the entire driver-vehicle-environment system [ROJ_11a]. Using the manoeuvre-building module integrated in its GUI as part of the present work, it is possible to generate arbitrary desired trajectories. This module uses road descriptions, user-defined functions or an integrated signal-creator to generate them.

The road is the main component of the environment. The developed road-building within the scenario-building module allows the creation of arbitrary roads. The integration of the open source program OpenCRG® makes possible the road generation. Synergies are used to add road perturbations and road roughness to the roads generated by OpenCRG® (without perturbations and roughness). Moreover, the scenario-building module can add sections with different friction coefficients to the road.

The CRG data structure of a road describes the road as a curved regular grid. The u-v coordinate system parameterizes the road as a rectangular grid. Thereby, u coordinate defines the reference line of the road and v coordinate locates those perpendicular road positions to the reference line. OpenCRG Visualization tools to show the road and OpenCRG evaluation tools to transform the road description from the *surface* to the *Cartesian coordinate system* and vice versa have been implemented in MOVES^2.

The animation module implemented in this work enables the visualization of the vehicle motion obtained from the simulation. This improves the movement analysis and the correlation between movement variables. Non-advanced users as well as experienced users benefit from the animation generation. The first user type may better understand how the vehicle interact along the simulation environment, while the second user type may use the animation to analyse the influence of more than one parameter at once.

The animation is generated through the integration of the Simulink 3D Animation™. Its VRML viewer displays the simulated scenario, whose appearance can be edited in order to highlight road perturbations and friction coefficient sections. Additional appearance configurations can paint lane lines on the road, can add any unanimated obstacle on the three-dimensional scene and can change

the vehicle shape. Additionally, a manual procedure can display several vehicles to compare their behaviour (for example, vehicles with different systems or components).

Available options can display static and dynamic text on a corner of the VRML viewer window and can show the real trajectory based on simulation results drawing a line on the road. Two different setups enable the animation generation at different moments during the simulation process: at the same time as the simulation execution or afterwards, when the simulation has finished. At any case the animation can be saved either as three-dimensional VRML format file or as two-dimensional AVI format file.

A procedure to set up a scenario to be simulated has been introduced in MOVES² in this work. A scenario is composed of road, manoeuvre and animation specifications.

The model management tool has been introduced in this work to handle the different model complexity levels MOVES^2 can deal with. Different model complexity levels are present in MOVES² at vehicle, system and component level. The system and component levels are represented by configurable subsystems: the *block choices*. Tables from a specific part of the GUI of MOVES² list all them and show their different available options. This characteristic feature of MOVES² enables the user to establish the required complexity level for his simulation purpose.

Further work may focus on the implementation of a management system for the complexity at vehicle model level. The model management tool implemented in this work sets the basis for this. In any case, the vehicle parameters (among others) must be adjusted in order to be able to use those parameters specified for a full vehicle model with other simpler models (e.g. mass point, quarter or single track vehicle model). Additionally, the procedure to visualize more than one vehicle during the animation can be automatized and a feature to animate interaction between the vehicle and the placed objects can be implemented. Another interesting work area could be the implementation of MOVES^2 on a real time platform and further on a driving simulator.

BIBLIOGRAPHY

[AAS_04] American Association of State Highway and Transportation Officials. *A Policy on Geometric Design of Highways and Streets.* Washington D.C., USA, 2004.

[AME_97] Ames, Andrea L.; David R. Nadeau and John L. Moreland. *VRML 2.0 Sourcebook.* Edited by Ryan Tim. John Wiley & Sons, Inc., 1997.

[ATK_02] Atkinson, Kendall. "Modelling a Road Using Spline Interpolation." Department of Computer Science, Department of Mathematics, The University of Iowa, Iowa, USA, 2002.

[CAS_96] Casey, James. *Exploring Curvature.* Department of Mechanical Engineering, University of California, Berkeley, USA: Friedrich Vieweg & Sohn Verlagsgesellschaft mbH, 1996, 113-136.

[CHA_10] Chaplier, Julien; and Philippe Ducloux. *RoadXML 2.2.1, Road Network Description: XML Format Specification.* France, 2010.

[DAI_10] Jochen, Rauh. "OpenCRG Workshop, Overview." DAIMLER, 2010.

[FIT_07] Fitzpatrick, Kay; Karl Zimmerman, Roger Bligh, Susan Chrysler and Byron Blaschke. "Criteria for high design speed facilities." PhD Thesis, Texas Transportation Institute, The Texas A&M University System, Texas, USA, 2007.

[FOR_98] Forkenbrock, Garrik; Mark Flick and W. Riley Garrot. "A test track performance evaluation of current production light vehicle antilock brake systems." *16th International Conference on Enhanced Safety of Vehicles.* Paper Nr. 98-S2-P-18. Windsor, Canada, 1998.

[FRÜ_04] Fründ, Jürgen; Jürgen Gausemeier, Carsten Matysczok and Rafael Radkowski. "Using Augmented Reality Technology to Support the Automobile Development." *Computer Supported Cooperative Work in Design I, 8th International Conference.* DOI. 10.1007/11568421_29. Xiamen, China: Springer Berlin / Heidelberg, 2004.

[GIA_04] Giaquinta, Mariano; and Stefan Hildebrandt. "Calculus of Variations I: The Lagrangian Formalism." 405-407. Germany: Springer, 2004.

[GIP_10] Gipser and Hofmann. *Cosin scientific software. Cosin/road, Cosin Road Modeling: Documentation and User's Guide.* Germany, 2010.

[GUS_98] Gustafsson, Fredik. "Monitoring Tire-Road Friction. Using The Wheel Slip." Department of Electrical Engineering, Linköping University, Linköping, Sweden, 1998.

[HIR_10] Hirz, Mario. “CAx in Fahrzeug- und Motorenbau.” *Lecture notes*. Institute of Automotive Engineering, Graz University of Technology, Graz, Austria, 2010.

[ISO_95] International Organization for Standardization. *Mechanical vibration: Road surface – Reporting of measured data.* Reference number: ISO 8608:1995, 1995.

[ISO_97] International Organization for Standarization. *Web3D Consortium, Open Standards for Real-Time 3D Communication.* 1997. http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97 (accessed September 8, 2011).

[ISO_99] International Organization for Standarization. *Passenger cars – Test track for a severe lane-change manoeuvre - Part 1: Double lane-change.* Reference number: ISO 3888-1:1999(E), 1999.

[KRA_11] Kraus, Harald. “Modellentwicklung zur Simulation erwarteter Bewegungstrajektorien von Fahrzeugen für eine Verkehrsflußsimulation.” Master's thesis, Institute of Automotive Engineering, Graz University of Technology, Graz, Austria, 2011.

[LIG_05] Ligos Corporation. *V·Realm Builder: User's Guide and Reference.* California, USA, 2005.

[MA_95] Ma, Weiyin; and J. P. Kruth. “Parameterization of randomly measured points for least squares fitting of B-spline curves and surfaces.” Great Britain, 1995.

[MAT_10] The MathWorks, Inc. *Simulink 3D Animation 5: User's Guide.* USA, 2010.

[MAT_11] The MathWorks, Inc. “Simulink 7: User's Guide.” 29_51, 29_85-29_93. USA, 2011.

[MIN_00] Ministerio de Fomento. *Norma 3.1-IC. Trazado, de la Instrucción de Carreteras.* Madrid, Spain: Boletín Oficial del Estado, 2000.

[NIE_10] Niederkofler, Haymo; and Andrés Eduardo Rojas Rojas. “Toward ecorner- energy efficient design of mechatronic suspension systems.” *FISITA 2010 World Automotive Congress.* Budapest, Hungary, 2010.

[NIE_11] Niederkofler, Haymo; Cornelia Lex, Arno Eichberger and Andrés Eduardo Rojas Rojas. “Potential Analysis of Active Suspension- and Traction Systems for Advanced Driver Assistance Systems in Critical Driving Situations.” *VDI Reifen-Fahrwerk-Fahrbahn.* Hannover, Germany, 2011.

[NOA_10] Stevenson, Angus; and Christine A. Lindberg, . *The New Oxford American Dictionary, Third Edition.* Oxford University Press, 2010.

[PAS_05] Pascual , Pere; Carles Bonet, Albert Compta, Neus Cónsul, Mercè Ollé and Agustí Roig. “Càlcul integral per a enginyers.” 87-114. Barcelona, Spain: Edicions UPC, 2005.

[ROJ_11a] Rojas Rojas, Andrés Eduardo; Haymo Niederkofler, Xavier Bas Ferrer and Josef Dürnberger. “Modular modelling of vehicles with innovative powertrain systems.” *European Automotive Engineers Cooperation, 13th European Automotive Congress.* Paper Nr. EAEC2011_C27. València, Spain, 2011.

[ROJ_11b] Rojas Rojas, Andrés Eduardo; Haymo Niederkofler and Wolfgang Hirschberg. “Mechanical Design of In-Wheel Motor Driven Vehicles with Torque-Vectoring.” *SAE Brazil Congress.* Paper Nr. 2011-36-0132. Brasil, 2011.

[SHI_95] Shikin, Eugene V. “Handbook and Atlas of Curves.” 407-429. Florida, USA: CRC Press, 1995.

[SIE_07] Sterbak, Rolf. “Lowering Emissions.” In *Pictures of the Future: The Magazine for Research and Innovation*, by SIEMENS, 99-101. 2007.

[VIR_10] VIRES Simulationstechnologie GmbH. *OpenCRG: User Manual.* Germany, 2010.

[VIR_11] VIRES Simulationstechnologie GmbH. *OpenCRG - managing the road surface ahead.* 27 May 2011. OpenCRG® tools (C-API and MATLAB) 1.0.2 available in Internet under: http://www.opencrg.org/ (accessed July 26, 2011).

[WAK_98] Wakool Shire Council. *Geometric Road Design (Urban and Rural).* Wakool, Australia, 1998.

A. THE GUI SCENARIO PANELS

Several panels of the GUI of MOVES^2, developed in this work, allow generating, modifying, deleting, and visualizing the information related to the simulation scenario (see Chapter 5); these panels manage with a high level of user-friendliness the road- and the obstacle-builder (see sub-chapters 3.3 and 3.4), the manoeuvre-building module (see sub-chapter 2.5) and the animation module (see Chapter 4). Moreover, the developed panels can modify the content of those libraries and sub-libraries from the *Parameters library* and those database-libraries that collect data related to the scenario (see sub-chapters 2.3 and 5.1).

According to the scenario modularity explained in sub-chapter 5.1, the *scenario master toolbar* distinguishes and gives access to the road, manoeuvre and animation specifications. As Figure A.1 shows, the *scenario master toolbar* displays on the one hand the selected scenario parameter script in MOVES^2 and on the other hand the scenario tabs.

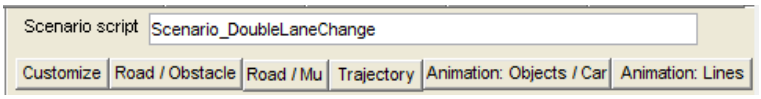


Figure A.1. The scenario master toolbar.

In order to choose, create, delete, and edit the parameter scripts from libraries and sub-libraries from the *Parameters library*, the generic and so-called *script management* sub-panel has been developed in this work. Figure A.2 illustrates this sub-panel. The top part of the sub-panel displays the name of the selected parameter file, in this case the *Scenario_DoubleLaneChange*. The bottom part of the sub-panel locates the buttons to manage the content of the library or sub-library – in this case the management is from a library, while a library collects the scenario parameter scripts. *New* button generates a blank and new parameter script from the library, *New from* button generates a new parameter script as well, but copying the data from an existing script, *Load* button loads the data from a file to the panel where this *script management* sub-panel is located, *Delete* button opens a list with the library content where the selected parameter scripts are deleted and *Edit* button loads the selected parameter script into MATLAB editor.

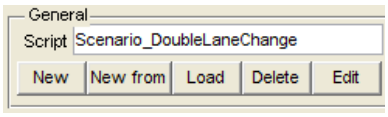


Figure A.2. Script management sub-panel to modify the content of libraries and sub-libraries from the Parameters library.

The *Customize* panel, shown in Figure A.3, is used for manage three different aspects of the simulation scenario. The top left part of the panel locates the *script management* sub-panel to manage the scenario library, as explained before. The top right part of the panel places the *Setup*

animation. It allows adjusting the animation performance described in sub-chapter 4.3 (to enable or disable the animation as well as to enable or disable the options to add predefined data and the real trajectory within the virtual world). The bottom part of the panel locates the *Script manager*, a tool to easily define the road, manoeuvre and animation specifications of the selected scenario from existing parameter scripts stored in their respective sub-libraries.

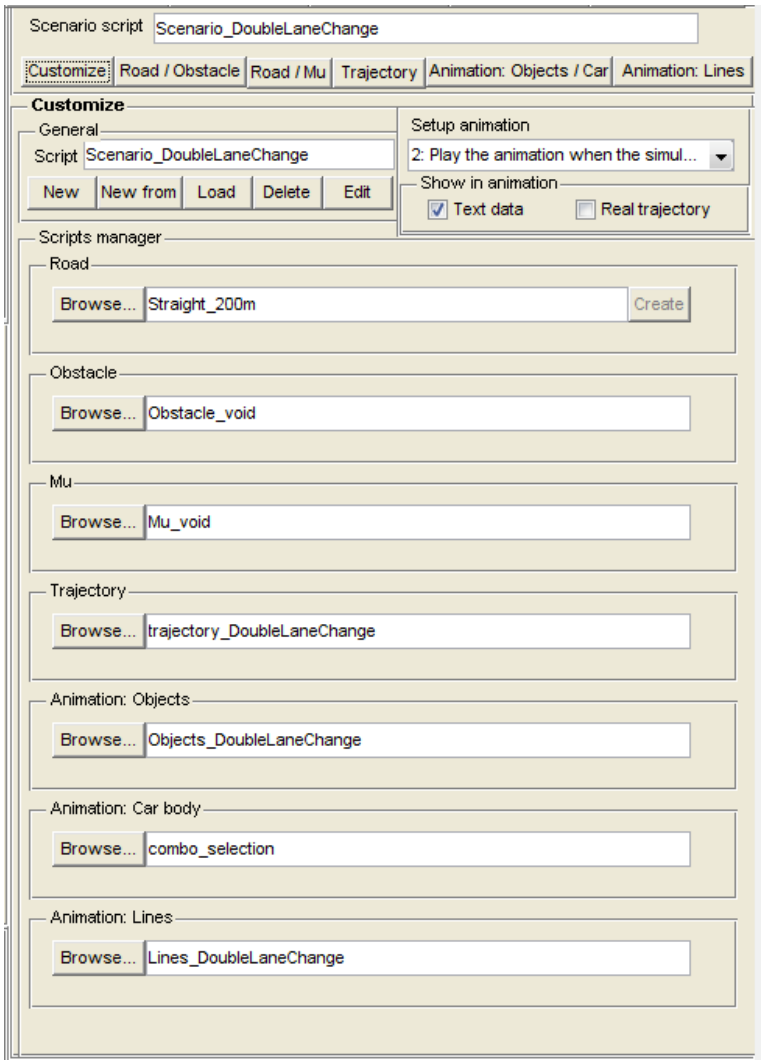


Figure A.3. Customize panel for a double lane-change manoeuvre.

The example shown in Figure A.3 is the *Scenario_DoubleLaneChange* scenario parameter script. It is configured to perform the simulation environment of a double lane-change manoeuvre [ISO_99]. It is composed of a 200 meters straight road with two lanes (*Road* and *Animation: Lines* specifications), the placement of cones on the road (*Animation: Objects* specification), and a predefined target trajectory (*Trajectory* specification).

Figure A.4 shows the *Road* / Obstacle panel. It is formed by three sub-panels and they define the road profile (see Chapter 3): The first sub-panel is the *Road* panel and it allows defining the road alignment while generating and visualizing the road anytime with 2-D and 3-D plots (see the tutorial in Appendix B). The second sub-panel, the *Obstacle* panel, performs the obstacle-builder. As explained

in sub-chapter 3.4.1, it can add road perturbations by selecting a 2-D perturbation profile from the corresponding database-library and defining the properties to extrude the perturbation profile. Moreover, each obstacle can have additional information regarding its appearance in the *scenario world* when the animation is enabled (see sub-chapter 4.2.2). They can be individually coloured using single colours or colour maps. The colour gradient from the colour maps is based on the elevation of the obstacle. The last sub-panel is the *Roughness* panel. It allows adding the road roughness following ISO standards [ISO_95]. The configuration from the example generates a bumped and straight test track without roughness.

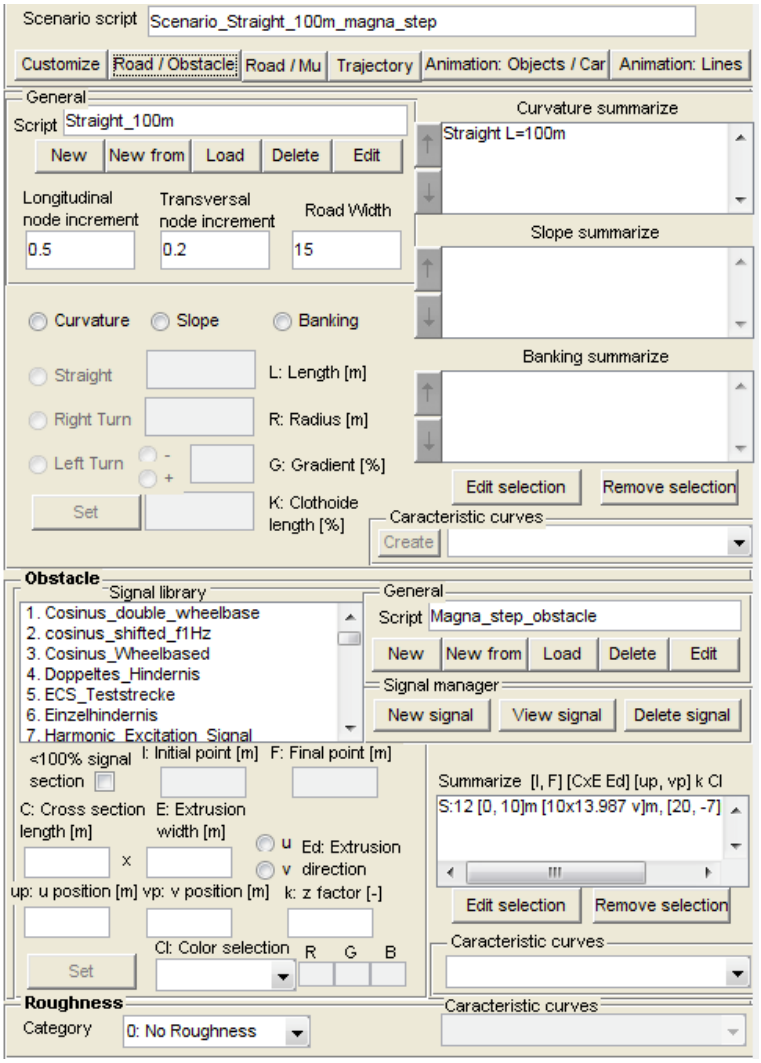


Figure A.4. Road / Obstacle panel for a bumped road.

Note that in the *Obstacle* panel, the *Signal manager* allows editing and visualizing the perturbation profile database-library by creating and deleting signals. New perturbation profiles can be generated using the Simulink block *Signal Builder* of MATLAB/Simulink® through the *New signal* button.

The next tab, the *Road I Mu* panel contains the *Road*, the *Roughness* and the *Mu* sub-panels, sharing the two firsts sub-panels with the *Road / Obstacle* panel. The *Mu* sub-panel can determine the road friction coefficient definition using road sections (see sub-chapter 3.4.3). Figure A.5 shows the *Road I*

Mu panel configured for a split-mu manoeuvre on a road with 0,8 and 0,3 friction coefficients side-to-side [NIE_11].

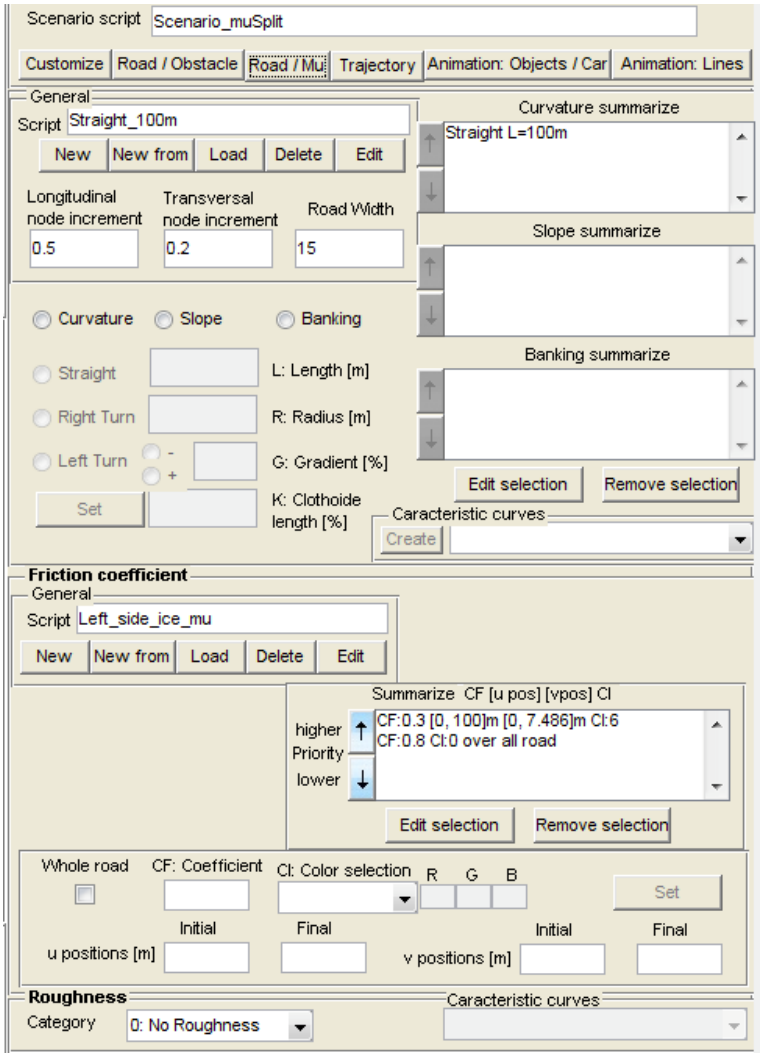


Figure A.5. Road / Mu panel for a split-mu manoeuver.

It is noteworthy that, in the *Mu* panel, if a road section associated to a friction coefficient overlaps another road section, the friction coefficient from the conflicted area is the one with more priority. The *arrows* from the *Summarize* sub-panel can assign the priority of the different friction coefficient definitions. Moreover, each friction coefficient definition can be coloured in the *scenario world* when the animation is enabled by adding additional appearance information (see sub-chapter 4.2.2 again).

When the parameter of the scenario parameter script links a road alignment parameter script whose CRG road is stored in the blank CRG road database-library (see sub-chapter 5.2), the three last taps from the *scenario master toolbar* are enabled. These taps open the *Trajectory*, *Animation: Objects / Car* and *Animation: Lines* panels together with the *schematic road plot* panel, which is common.

Figure A.6 shows the *Trajectory* panel. This panel performs the manoeuvre-building module. As in any case, the trajectory sub-library is managed by the *script management* sub-panel, which is located

on the top right part of the *Trajectory* panel. The central right part of the panel allows specifying an extra offset to the defined trajectory and the option to show the desired trajectory in the animation when it is enabled. If the desired trajectory is defined out of the road boundaries, the box to add the offset automatically turns red. The rest of the panel is variable according to the option selected in the *Source* sub-panel. Three options are available to generate the desired trajectory as explained in sub-chapter 2.5: the roads from the blank CRG road database-library, a trajectory from the trajectory database-library and user-defined functions.

The example belongs to a double lane-change manoeuvre. In this case the target trajectory is set without any additional offset by the *Trajectory_DoubleLaneChange_mat* trajectory from the trajectory database-library. A green custom colour determines this desired trajectory in the *scenario world*.

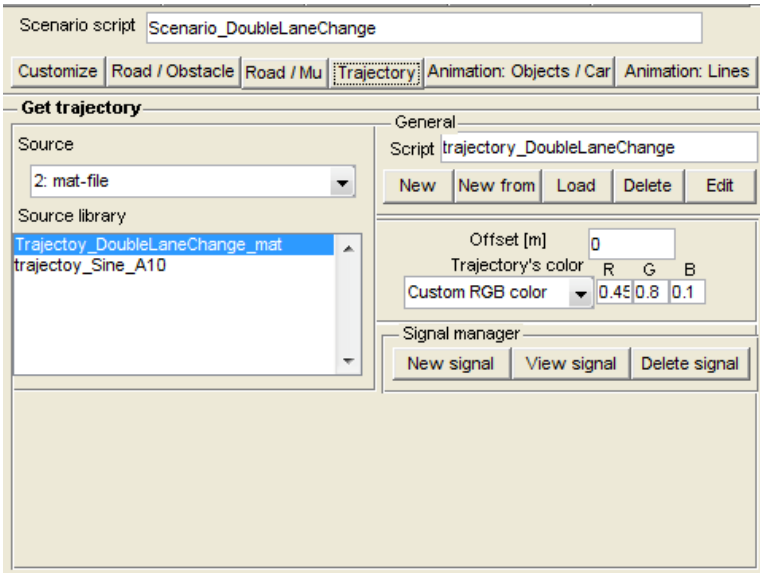


Figure A.6. Trajectory panel for a severe double lane-change manoeuvre.

Note that the second option of the manoeuvre-building module enables a list box and the *Signal manager* sub-panel. The list box displays the data from the trajectory database-library, while the *Signal manager* edits its content. *New signal* button can create new user-defined signals by using the *Signal Builder* tool of MATLAB/Simulink®. This new signals are stored in the trajectory database-library. *View signal* plots the selected trajectory in the list box. *Delete signal* removes the selected trajectory from the corresponding database-library.

The *Animation: Objects / Car* panel together with the *Animation: Lines* panel configure the animation specifications (see sub-chapter 4.2).

Figure A.7 shows the *Animation: Objects / Car* panel. It allows managing the objects and the vehicle shape parameter scripts. The bottom right part of the panel locates the *Car body* panel. Through the pop-up menu it is possible to select the vehicle shape for the animation by choosing one VRML vehicle shape from the vehicle shape VRML database-library. The rest of the panel configures the *Objects* panel. The objects can be added individually using the *Object library* list box and fulfilling the

position and rotation specifications, or can be added in groups using the *Script library* list box and fulfilling the initial position for the group of objects. The *Object library* list box displays the content of the unanimated objects VRML database-library, while the *Open object* button opens the selected object using the Ligos V·Realm™ Builder software [LIG_05]. The *Script library* list box displays the parameter scripts of the unanimated objects sub-library.

The example shown in Figure A.7 corresponds to double lane-change manoeuvre. The objects parameter script configures the necessary cones to define the manoeuvre while the vehicle shape parameter script selects the animated vehicle shape.

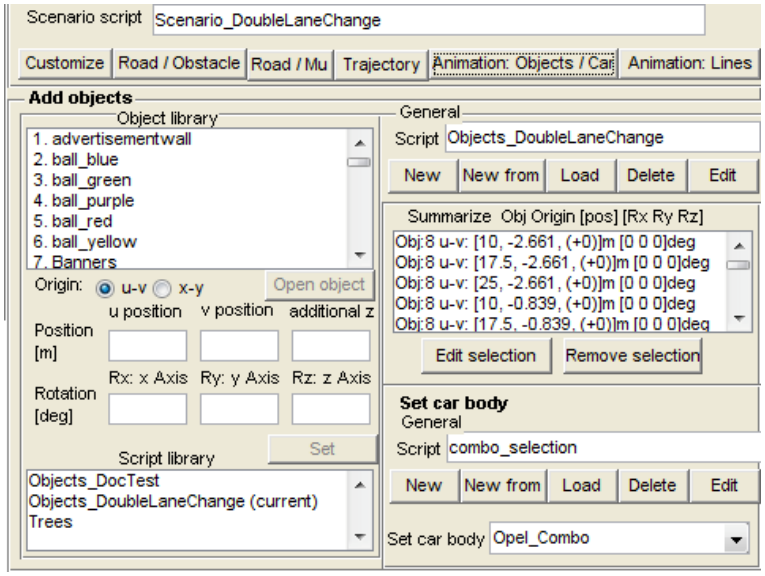


Figure A.7. Animation: Objects / Car panel for a double lane-change manoeuvre.

Figure A.8 illustrates the *Animation: Lines* panel. It defines the lane lines parameter scripts. As stated in sub-chapter 4.2.2, within the *scenario world* the lane lines may be defined. To definition of one lane line requires its v position and its *width* as well as its *line length* and *gap* in case it is composed of dashes. *Edit selection* button can edit the selected lane lines from the *Summarize* list box while the *Remove selection* button removes it.

The example of Figure A.8 is the lane lines parameter script for a double lane-change manoeuvre. It defines a dashed line on the road midline and two lines to mark the road limits.

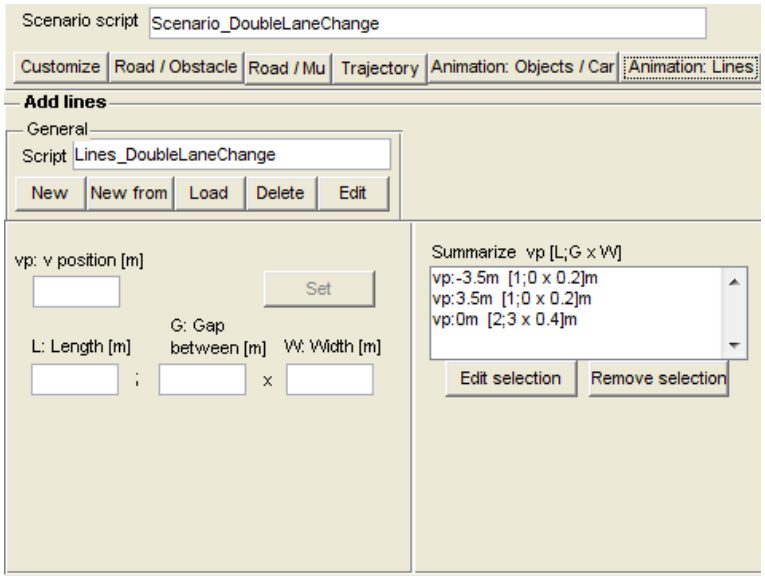


Figure A.8. Animation: Lines panel for a double lane-change manoeuvre.

Figure A.9 illustrates the *schematic road plot* panel shared by the *Trajectory*, *Animation: Objects / Car* and *Animation: Lines* panels and configured for a double lane-change manoeuvre. As it shows, check boxes allow the addition of the manoeuvre and animation specifications on the plotted road. In this panel the already created road is visualised in *Cartesian coordinates* through its edges, and through the corresponding check boxes it is possible to enable and disable the target trajectory, the selected objects, and road lines.

Figure A.9. The schematic road plot panel for a double lane-change manoeuvre.

B. THE GUI ROAD PANEL

The current pages describe the usage of the *Road* panel from the GUI of MOVES^2 to generate roads with a specific vertical and horizontal alignment (see Chapter 3). Figure B.1 illustrates this panel.

As the figure shows, the *Road* panel is divided into four parts:

The top left side part shows general data from the selected parameter script and it is split into two sections. Its upper section locates the *script management* sub-panel, defined in the Appendix A, to choose, create, delete, and edit the parameter scripts from the road alignment sub-library (see Chapter 5). Through its lower section general data concerning the road can be edited: two parameters establish the longitudinal and transversal road grid discretization as the grid node increment in u and v direction respectively, and another parameter defines the road width. It is noteworthy that the selection of *Road_void* road alignment parameter script, the empty parameter script, disables almost the whole panel. This is the case of Figure B.1.

The bottom left side part of the panel places a compact set of buttons to introduce the road alignment (see sub-chapter 3.2). *Curvature* button can set horizontal alignment road sections of *tangents*, *curves* and *transition curves*. *Slope* button is in charge to set the vertical alignment: the *grade* or *slope*. *Banking* button can define the *superelevation* or *banking* horizontal alignment property using road sections. Each of them enable different buttons and edit boxes and some boxes are used to introduce parameters from different specifications. For this reason the explanation of how to define the road alignment continues further on with an example.

Three list boxes are placed on the top right hand side of the panel. They summarize the data introduced with the compact set of buttons. Each list box displays a line for every defined road section of the property shown in the list box. If the user selects any displayed line, the *Edit selection* button will transfer the data of the selected line to the set of buttons, and the *Remove selection* button will remove the selected road section from the alignment.

The bottom right side of the panel places the road visualisation tool. After setting the road alignment *Create* button generates a CRG data structure based on the current road definition, the so-called *blank CRG road* (see sub-chapter 3.4). Then the pop-up menu shows the visualisation available options according to the alignment. It is possible to plot the following: reference line map, reference line elevation, elevation grid cuts and limits, elevation grid surface without slope and banking, and road surface.

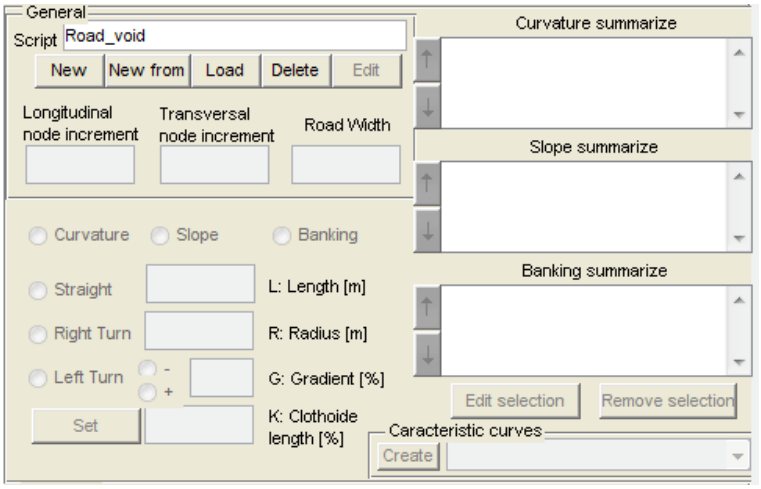


Figure B.1. The GUI Road panel with the void road file selected.

The next example shows step by step a road alignment generation. Assuming the previous calculus explained in sub-chapter 3.2, the test road will turn left after few meters downhill drawing a curve with *banking* (to aid the vehicle steer) of 20 meters radius and almost 2 meters of *transition curve*.

The first step is to create a new road alignment parameter script and to fulfil the road name and the general road parameters. Then, it may be necessary to click on *New* button to open a dialog box and save the new parameter script. After specify the general data of the road the panel may be as follows:

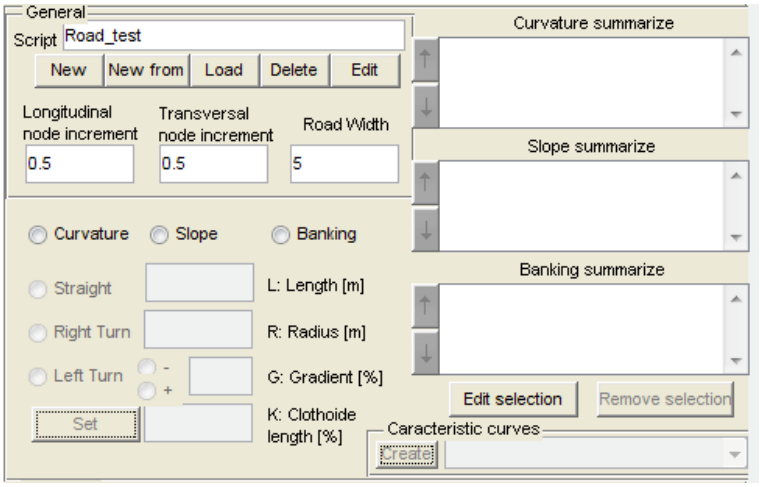


Figure B.2. Example of a road alignment construction. General data fulfilled.

Note that after create this new road alignment parameter script several buttons have enabled, thus the current selected parameter script is different from the *Road_void*.

Now it is possible to set a road section of either *curvature* or *slope* or *banking*. The best choice may be to start with the *curvature* definition. Figure B.3 shows the enabled button after select the *Curvature* button. If the user clicks on *Straight* just the *Length* edit box is enabled, while if he clicks on *Right Turn* or *Left turn* every edit box is enabled but *Gradient* box. After type the necessary data it is

possible to click on *Set* button to save and send the information to its list box summarizer. *Clothoide length* is a not necessary input at any case.

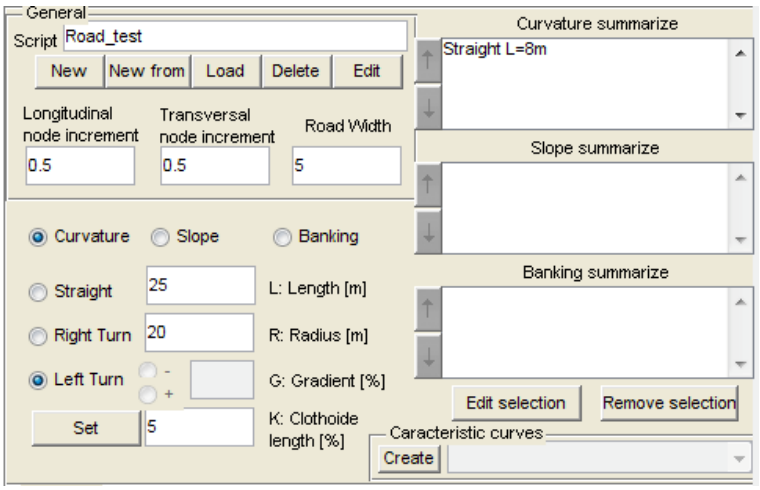


Figure B.3. Example of a road alignment construction. Introducing curvature definition I.

Note the previous road has already specified a straight section, so after add that section (see Figure B.4) it is possible to move up or down the selected rows, edit them or remove them. *Create* button generates the CRG structure data of the current road and it enables the pop-up with three available plots. Figure B.5 shows the subplot from *Road surface* characteristic curve, where the road is plot in *Cartesian coordinates*.

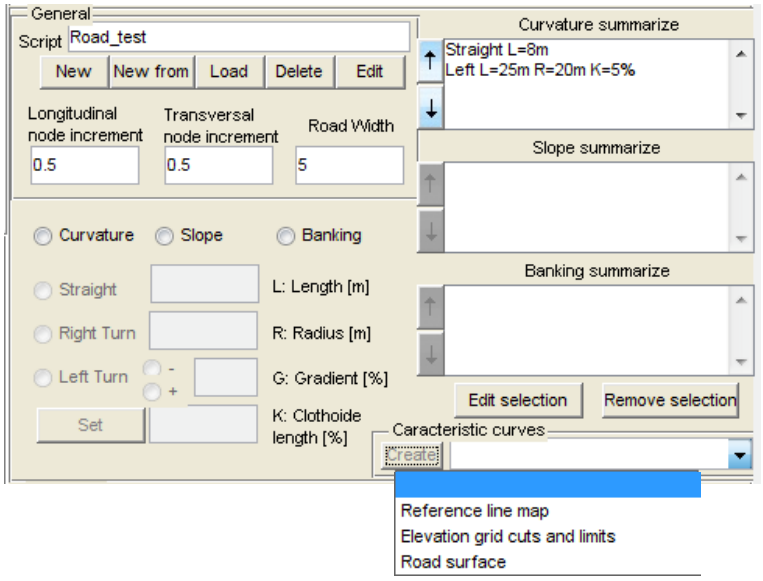


Figure B.4. Example of a road alignment construction. Introducing curvature definition II.

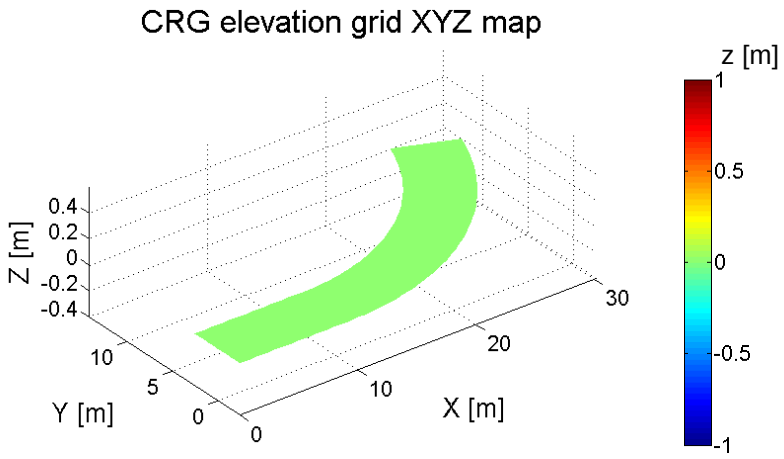


Figure B.5. 3-D plot of the road example. Curvature specified.

The procedure to add *banking* is similar to the previous one. This time, *Banking* button enables *Length*, *Gradient*, and *Clothoide length* edit boxes, and the - and + buttons. These radio buttons define the road side to elevate, according to the *right-hand rule* (thumb on the road midline pointing from low u values to high u values). In the example, - is pushed, so as Figure B.7 shows the right side of the road has more altitude compared to the left road side. Note that defining a blank section (to generate a blank section, fulfil only *Length* box) it is possible to define a *banking* section after n meters. One consequence of generate a road with *curvature* and *banking* is that more plots are available in the pop-up menu (see Figure B.6).

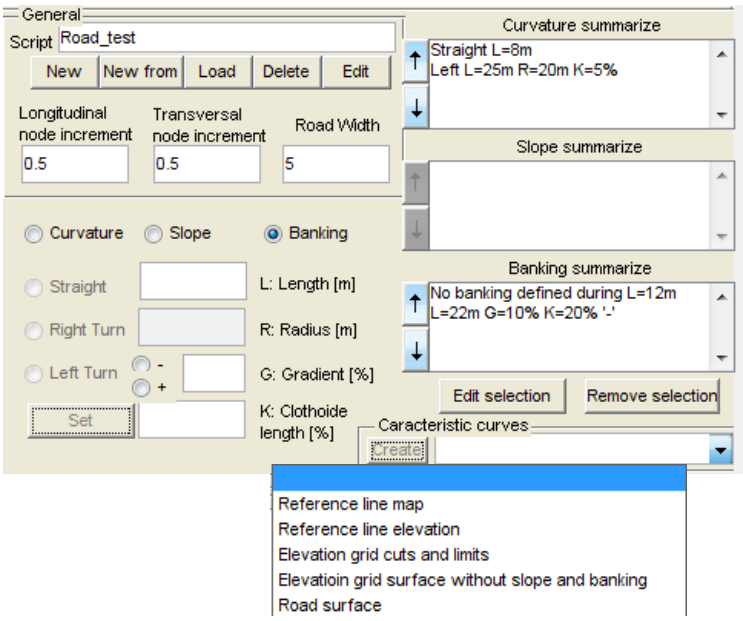


Figure B.6. Example of a road alignment construction. Introducing banking definition.

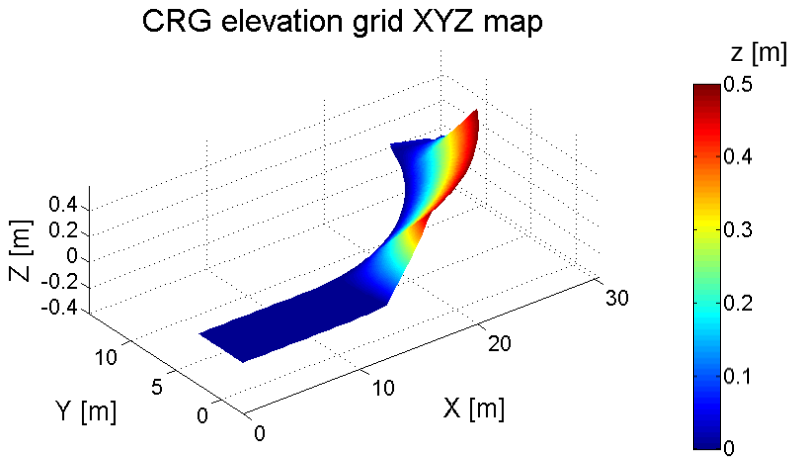


Figure B.7. 3-D plot of the road example. Curvature and banking specified.

Finally, to set the initial ramp on the road, a *slope* is add during the first 5 meters. Figure B.8 shows the slope specifications and Figure B.9 illustrates the result. Note that *Slope* button enables *Length*, *Gradient*, and *Clothoide length* edit boxes.

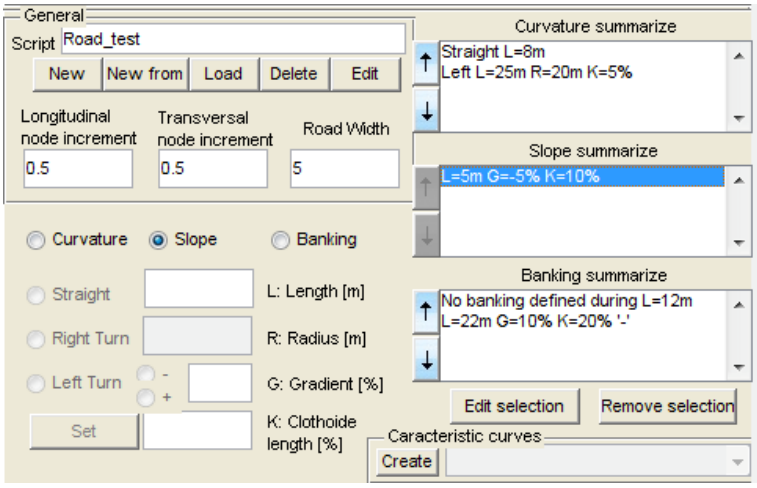


Figure B.8. Example of a road alignment construction. Introducing slope definition.

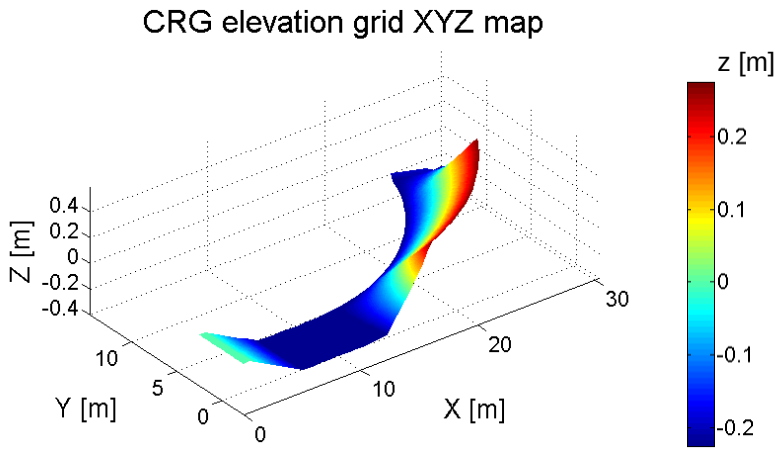


Figure B.9. 3-D plot of the road example. Curvature, banking, and slope specified.

C. EXPORT CATIA MODELS TO THE VRML FORMAT

An easy way to obtain VRML object files is to export them from CAD designs. For example CATIA® V5, software commonly used in engineering, has the tool to convert their models into VRML format file. The current pages describe and include some tips regarding how to export CATIA parts to the VRML format.

To export CATIA parts to the VRML format select *File* > *Save as* and chose VRML from the *Save as type* list, but before that, take into consideration some remarks:

It is possible to adjust some properties to improve the conversion. On the one hand CATIA® is capable to adjust the visualization mesh detail. To control the mesh accuracy, in the menu, select: *Tools* > *Options* > *General* > *Display* > *Performances*. The perfect setting creates the smallest possible VRML file that performs acceptable appearance. On the other hand export filter settings are available on *Tools* > *Options* > *General* > *Compatibility* > *VRML*. Figure C.1 illustrates the panel to adjust filter settings. Select the export format VRML97 while it is the used in Simulink 3D Animation™. Select the *Save normals* check box and clear *Save edges* check box for better performance. Finally, select the appropriated *Save textures* check boxes. Actually it is possible to generate external JPG files of the textures from the current model, to get them select *Save textures in external files* option [MAT_10].

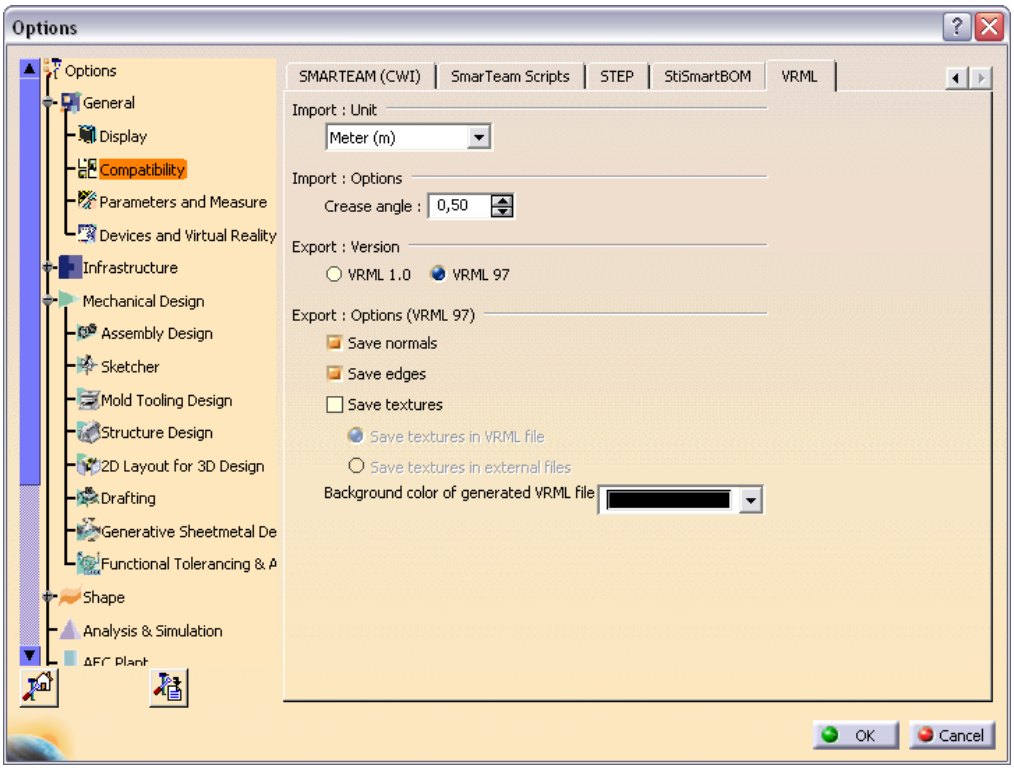
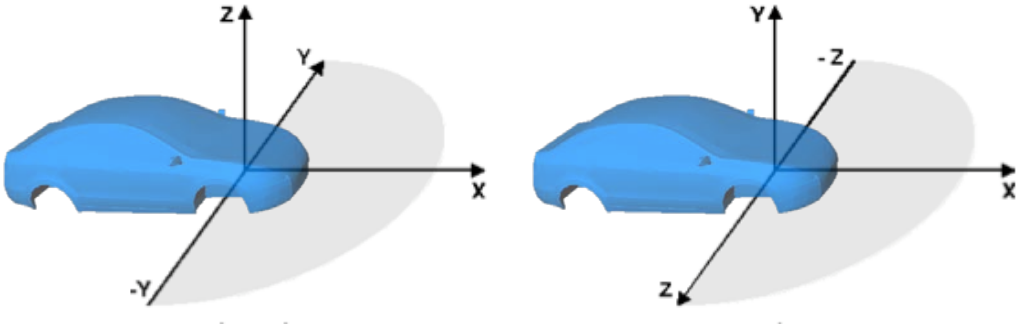


Figure C.1. VRML Compatibility dialog box.

The following figure is a reminder about the coordinate system handled in Simulink 3D Animation™ and the one used in CATIA® V5 (see sub-chapter 4.1.1). Note that the axis corresponding to the driving direction for a vehicle shape is in both cases the x axis.



CATIA coordinate system VRML coordinate system

Figure C.2. CATIA and VRML coordinate systems [MAT_10].

One solution to export the CATIA part with the good orientation is to turn it over the appropriate axles to change its coordinate system. Then rotate the part -90 degrees over x axis. The result should be as the shown in Figure C.3.

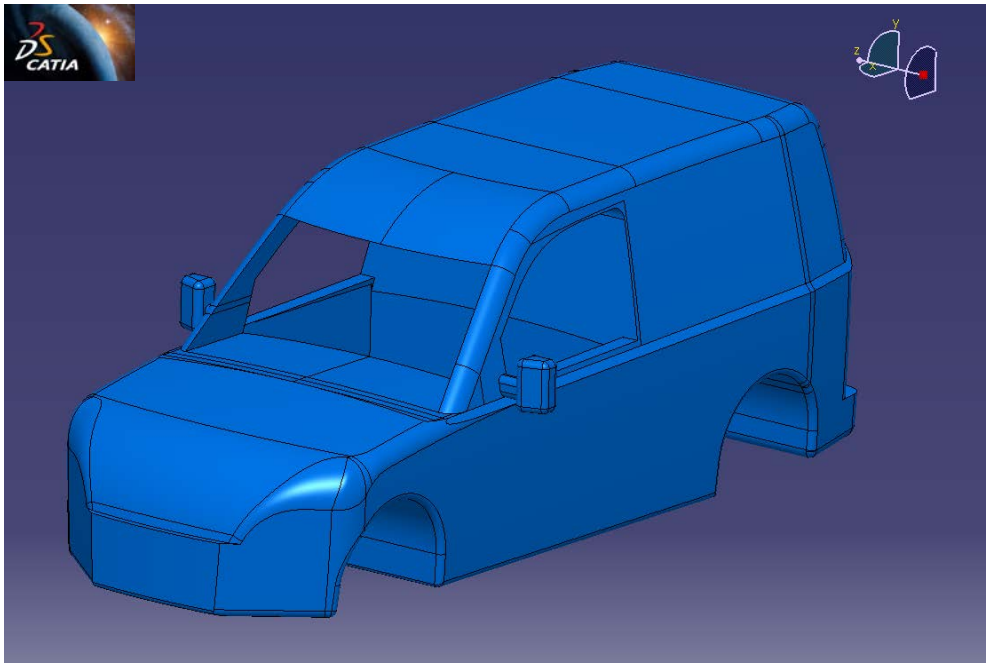


Figure C.3. CATIA standard view of the part, which is ready for the exportation.

If Ligos V·Realm™ Builder software opens the exported file, the preview would be as the shown in the following figure.

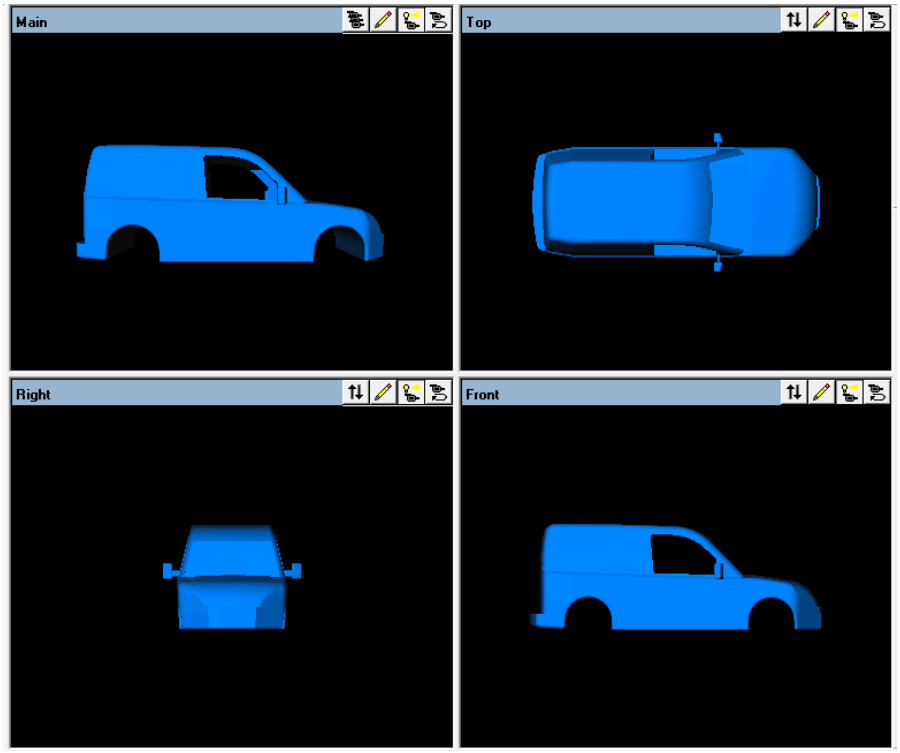


Figure C.4. The preview of Ligos V·Realm Builder for the exported object.

Figure C.5 shows the animation performance of MOVES[2], using the exported file as the vehicle shape (see sub-chapter 4.2).

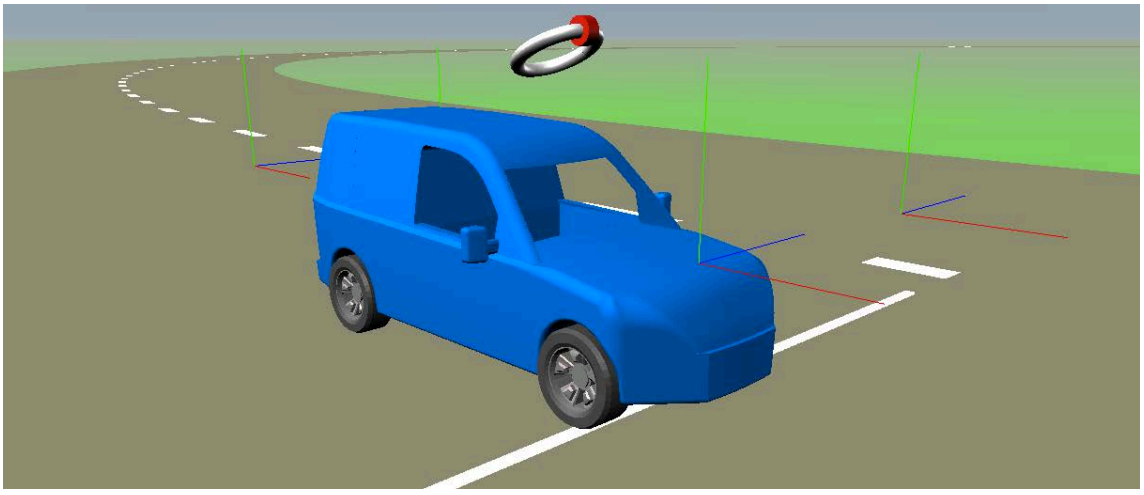


Figure C.5. The exported CATIA model working within the VRML scene. Screenshot of the visualization feature implemented in this work.

D. ANIMATION WITH MULTIPLE CARS

The content of the present appendix describes how to add the vehicle motion of a vehicle simulation into the animation file from another simulation (see Chapter 4). The description can be extended for n vehicle simulations.

First, use the animation module and select the setup that saves animations into VRML format file (see sub-chapter 4.3). Simulate twice to generate two animations with the same scenario definition but different simulation specifications.

Open with the MATLAB editor or other text editor one animation file – usually called *car_motion_scene_anim*. The structure of the animation file has a first part of definitions: geometry wheels, markers shapes (if the real trajectory is displayed), and the scenario world file (with its linked files) (see sub-chapter 4.2.2).The second part contains vehicle motion data. Note that the linkage using VRML language is executed with the command *url* followed by the path of the linked file (see sub-chapter 5.2).

Replace the word *car1* with the word *car2* (for example), and change the name of the linked vehicle shape.

Duplicate the vehicle shape file and rename it as the same name set at the previous step. Open the renamed file and change its appearance colour to distinguish it from the other vehicle. In the VRML code, RGB colour model manages colours of the virtual world. They are specified with the command *diffuseColor* inside the *Material* node, or with the command *color* inside the *Color* node.

Finally, copy the whole car definition from the edited animation file and paste it in the second animation file, just below its car definition. Copy the vehicle motion data (except markers) from the first animation file to the second animation file below the last line containing vehicle motion data.

Now the second animation file is ready to show the animation with both vehicles as in the Figure D.1, just type *vrplay* followed by the path of the file you want to watch on the *Command Window* of MATLAB.

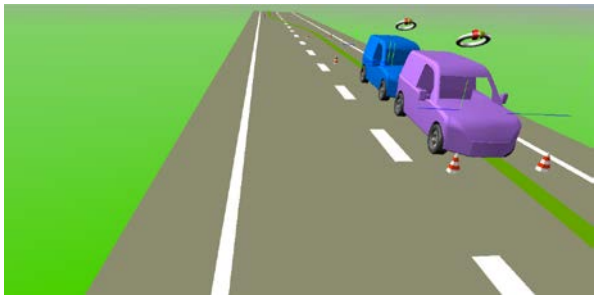


Figure D.1. Example of an animation with two cars on scene. Screenshot of the visualization feature implemented in this work.